



ALBERT-LUDWIGS-  
UNIVERSITÄT FREIBURG

# Algorithms for Radio Networks

## Data Aggregation

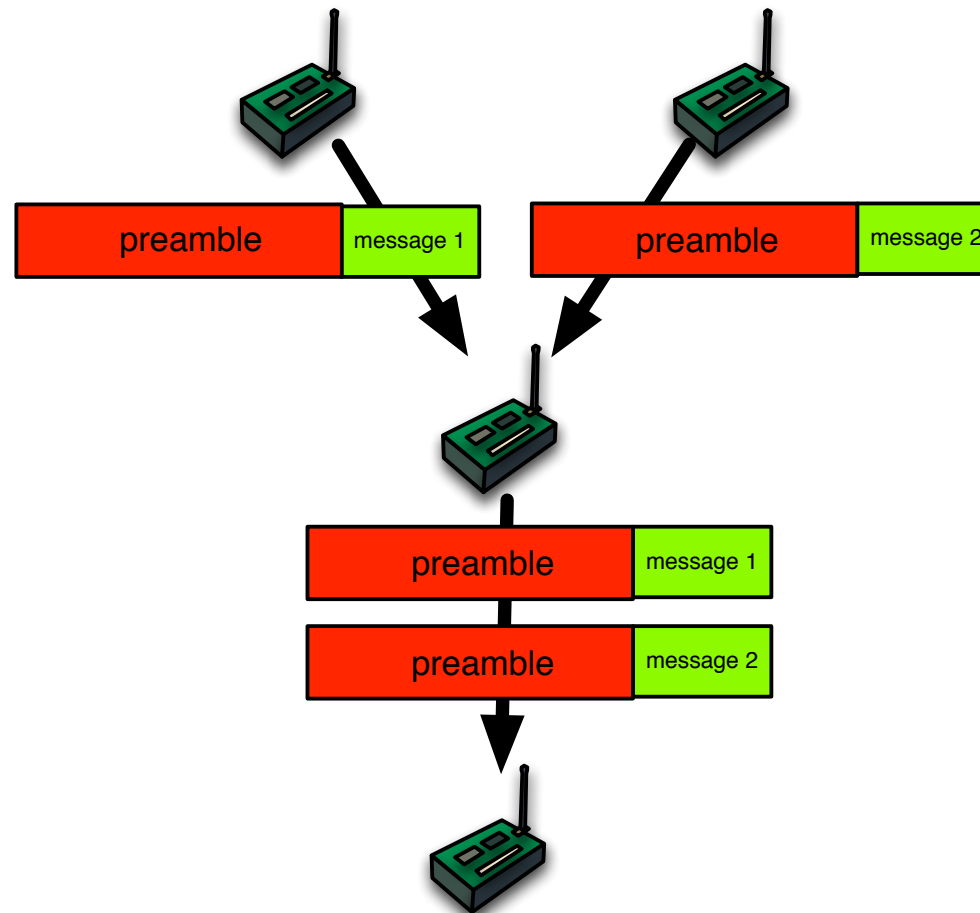
University of Freiburg  
Technical Faculty  
Computer Networks and Telematics  
Christian Schindelhauer



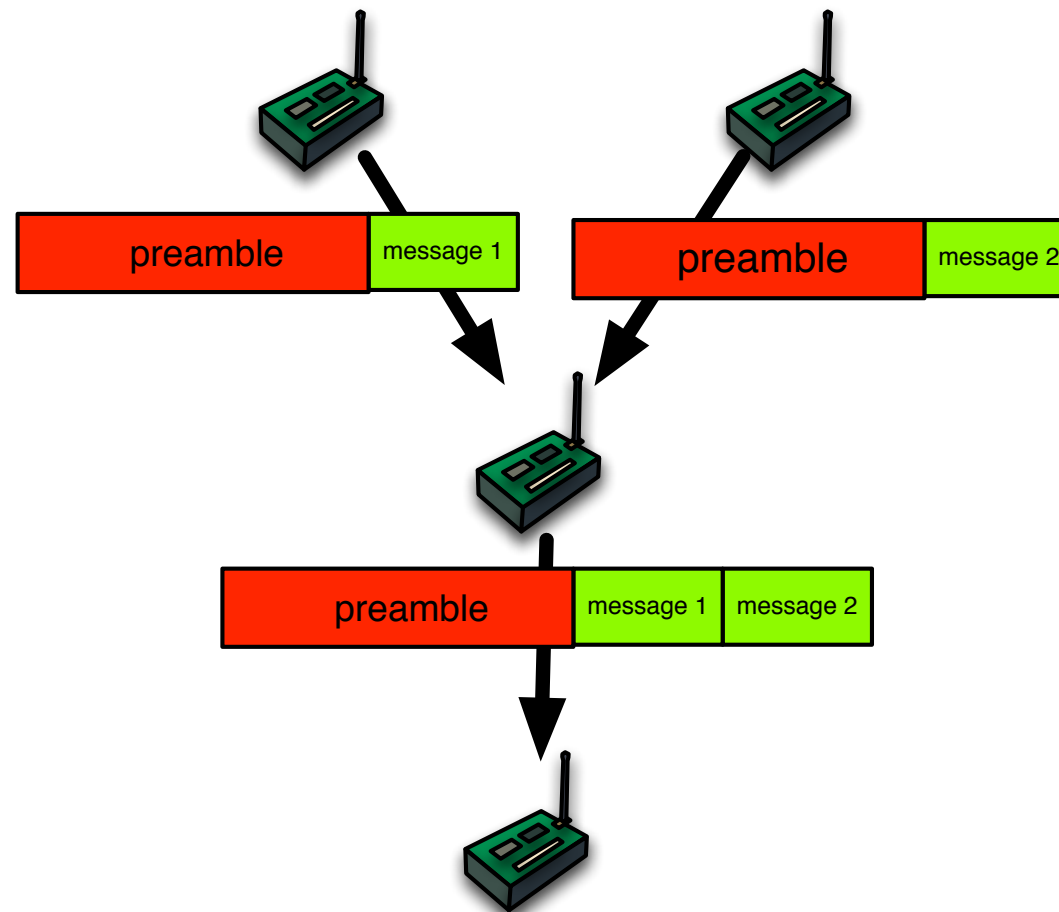
# Data Aggregation

- ▶ **In multi-hop networks combining message can improve networking**
- ▶ **Concatenation) of messages**
  - overall number of headers is reduced
    - especially for Preamble Sampling
  - smaller costs for collision avoidance
- ▶ **Recalculation of contents**
  - e.g. If the minimum temperature is required, then it satisfies to forward the smallest value
  - For this purpose, collect the input over some time

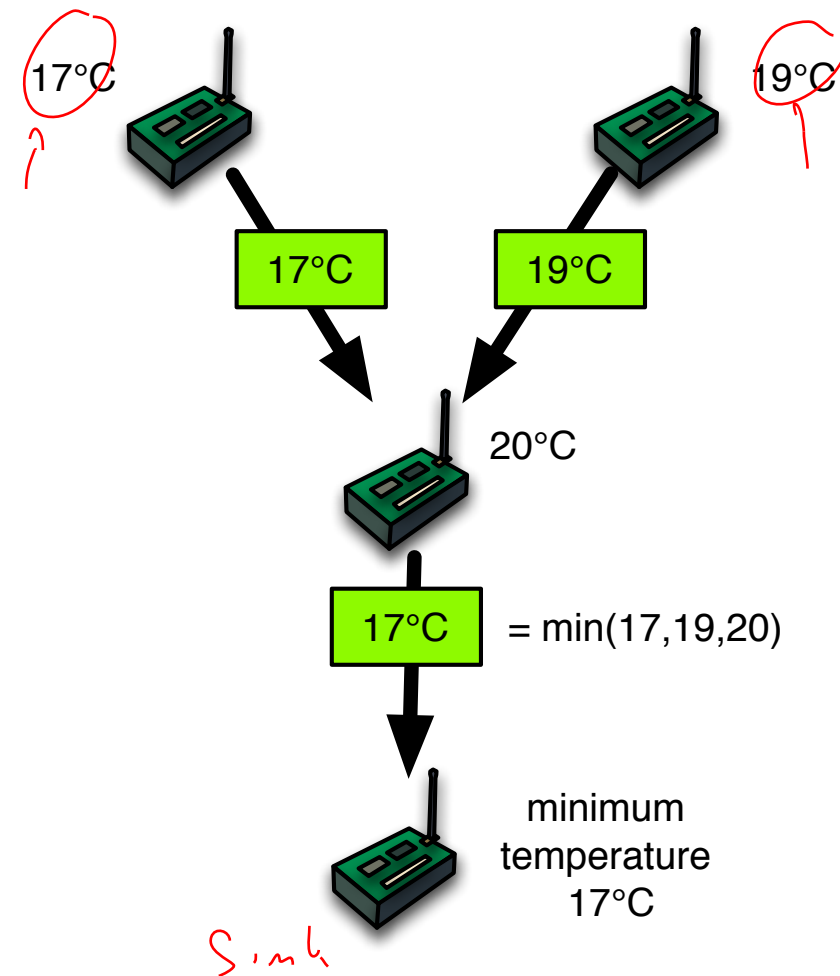
# No Data Aggregation



# Data Aggregation by Concatenation



# Real Data Aggregation by Recalculation



# Simple Functions for Data Aggregation

- ▶ Minimum
  - inner node computes the minimum of input values
- ▶ Maximum
  - like Minimum
- ▶ Number of sources
  - inner node adds input values
- ▶ Sum
  - addition at inner nodes

# Aggregable Functions

## ► Mean

- compute the number of sensors:  $n$
- compute the sum of sensor values:  $S$
- $\text{mean} = S/n$

## ► Variance

- Compute average and the average of squares of values
- $V(X) = E(X^2) - E(X)^2$

# Non-Aggregable Functions

- ▶ **The following functions cannot be aggregated easily**
  - median
  - p-quantile
    - if p is not very small or large
  - number of different values
    - only for large data sets an approximation is possible
- ▶ **Approximate solution**
  - was presented in „Medians and Beyond: New Aggregation Techniques for Sensor Networks, Shrivastava et al. Sensys 04
  - using k words in each message an approximation ratio of  $\log n/k$  can be achieved

# Routing Models for Data Aggregation

## ▶ **Address Centric Protocol**

- each sensor sends independently towards the sink
- not suitable for (real) aggregation

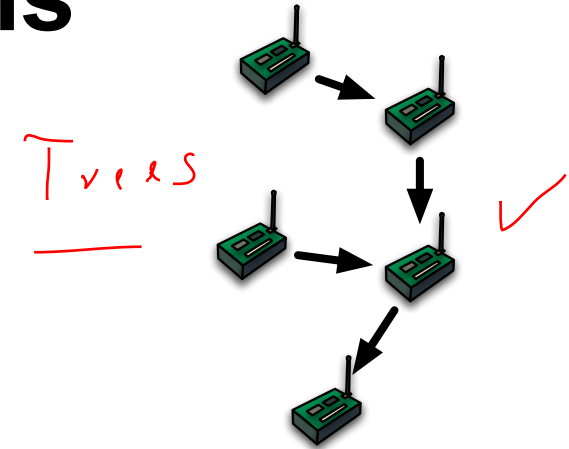
## ▶ **Data Centric Protocol**

- Forwarding nodes can read and change messages

# Communication Graphs

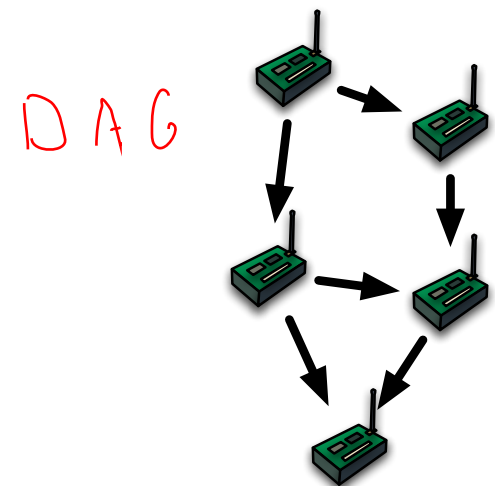
## ▶ Tree Structure

- If there is only a single sink
- and every source uses only a single path
- then every communication graph in a WSN is a tree



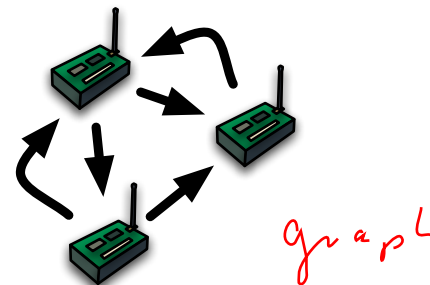
## ▶ DAG (directed acyclic graph)

- general case
- caused by changing routing paths to the sink
- may complicate data aggregation
  - e.g. sum



## ▶ General graph

- Population protocols
- are not used in WSNs



# Energy Optimal Tree Structure

► **Given:**

- set of data sources and a sink
- communication graph  $G$

► **Compute:**

- Steiner tree  $T$ 
  - sub-graph of  $G$
  - connects all sources and sinks
  - number of edges is minimal

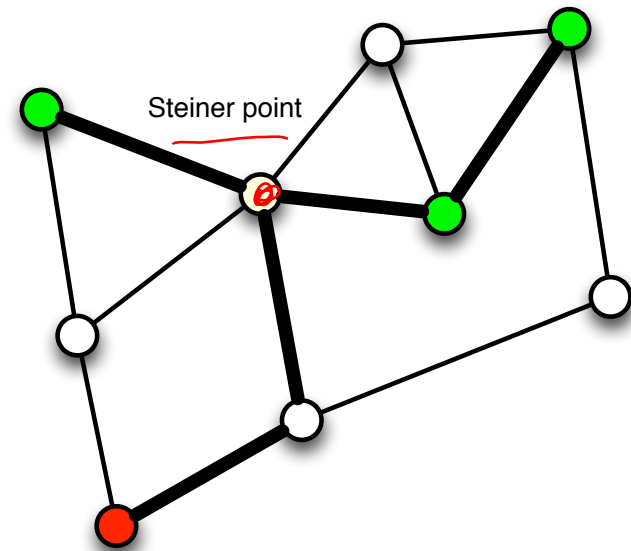
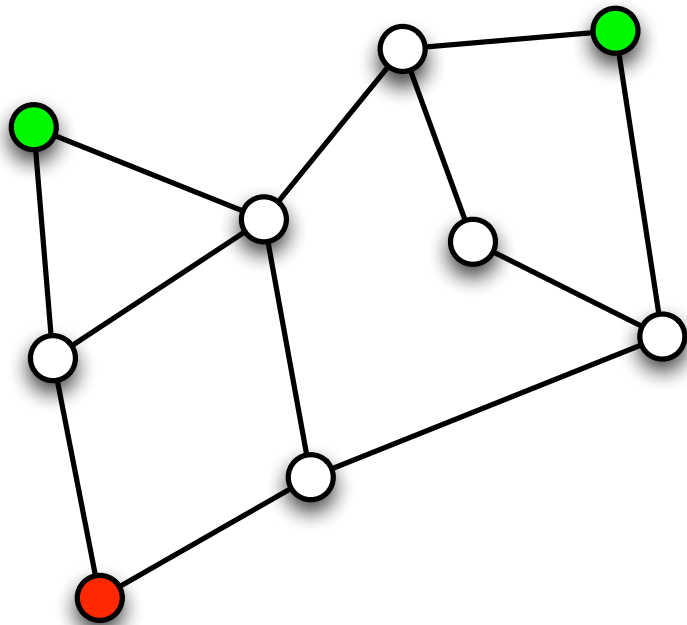
► **Alternative:**

- edges have an (energy) weight
- minimize the sum of edge weights

# Steiner Tree Problem

## ► Observation:

- sources and sinks can be handled the same way



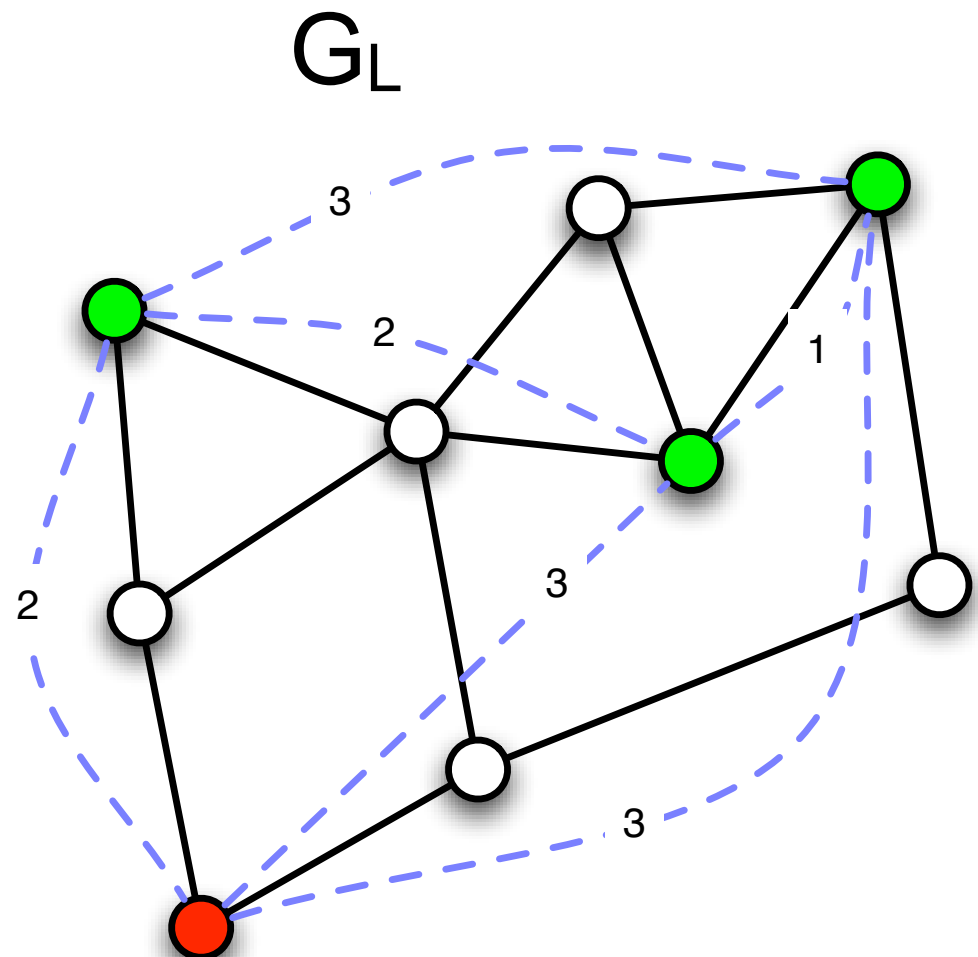
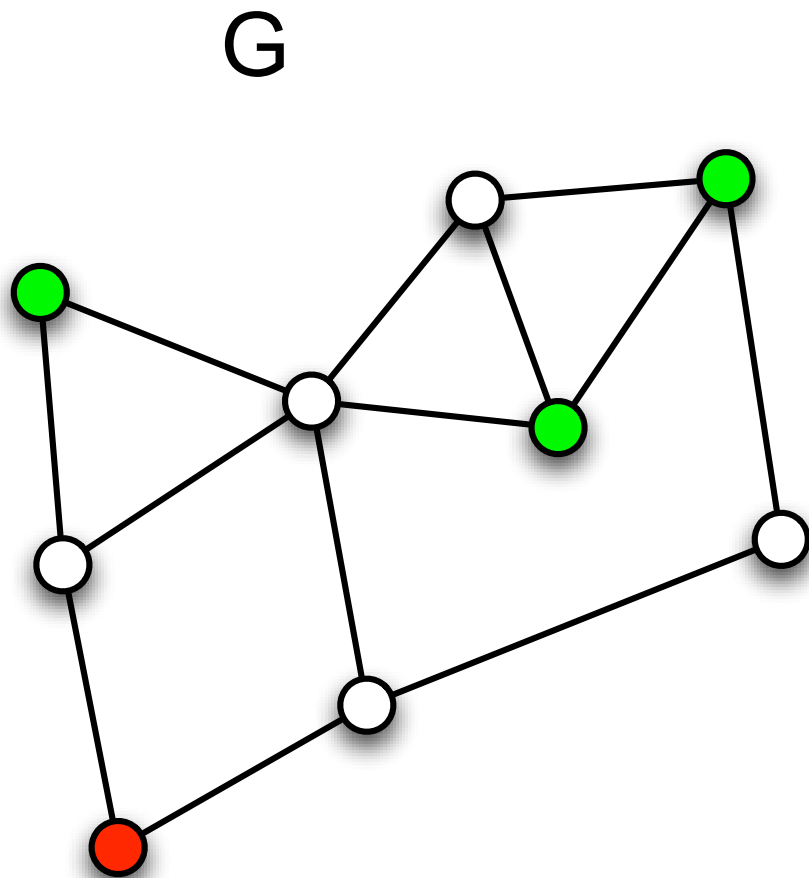
# Optimal Choice of Data Aggregation

- ▶ **Steiner tree**
  - minimal tree connecting all sources and sinks
- ▶ **Computation of the Steiner tree is NP-hard**
- ▶ **Approximation**
  - the Steiner tree problem can be approximated with factor 2
    - if the underlying graph is metric
  - best known approximation factor for algorithms in polynomial time: 1.55
    - Zelikovsky, Robins 2006

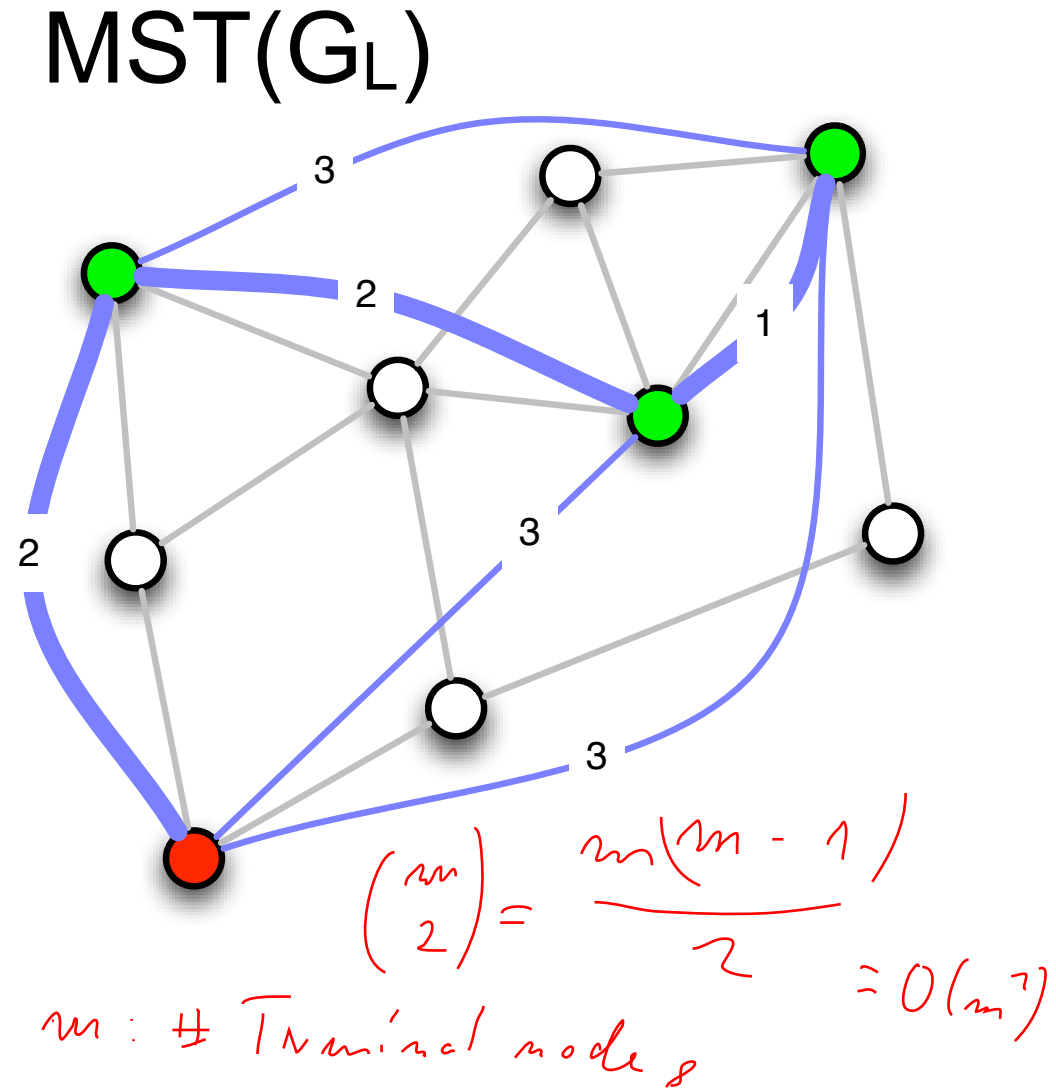
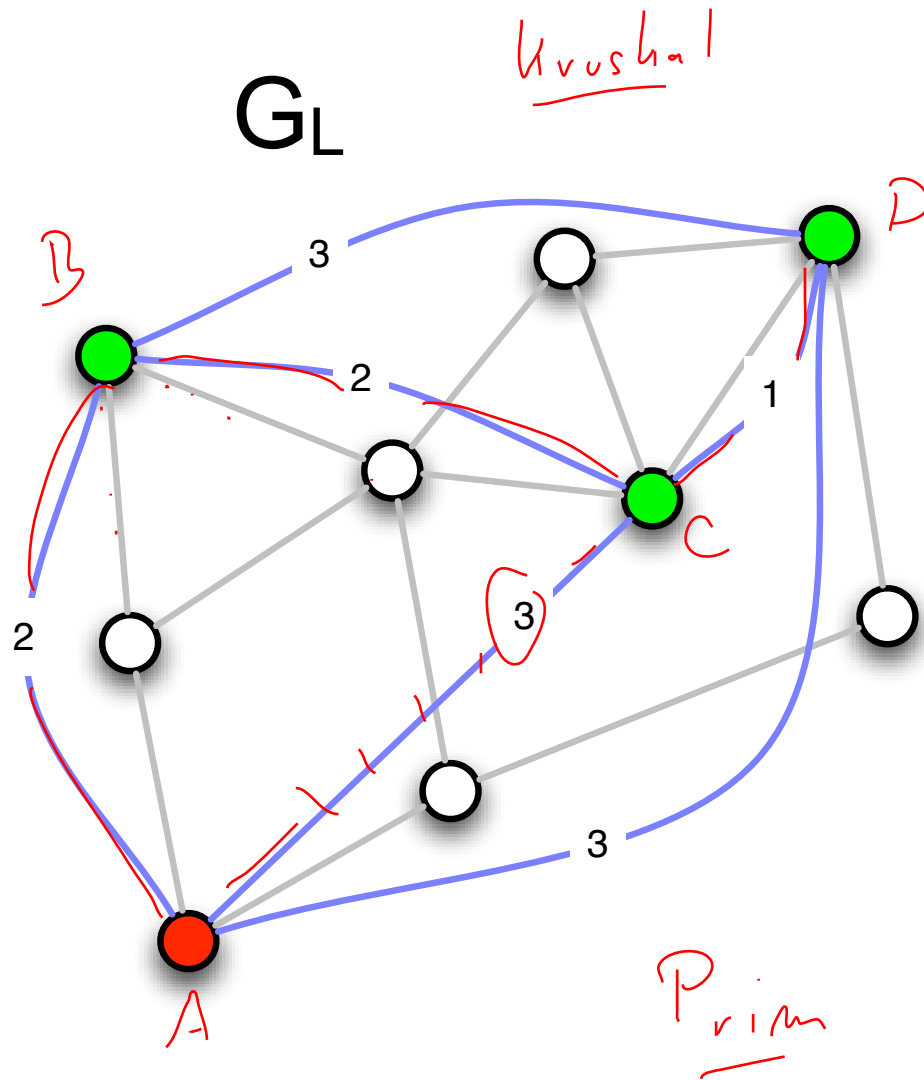
# Approximation with the Help of MST

- ▶ Source:
  - Information Processing Letters, 27 (1988), 125-128, Kurt Mehlhorn, *A Faster Approximation of the Steiner Tree Problem*
- ▶ Compute the distance between the terminal nodes in the graph  $G$ 
  - Compute the complete graph  $G_L$  with terminal nodes  $V_T$  and edge weight according to the distance in  $G$
- ▶ Compute **minimum spanning tree** in  $G_L$
- ▶ Initialize tree  $T$  with empty set
- ▶ For each edge  $e=(u,v)$  of the **MST**
  - Find shortest path  $P$  from  $u$  to  $v$  in  $G$
  - If less than two nodes of  $P$  are in  $T$  then
    - Insert  $P$  into  $T$
  - Else
    - Let  $p$  and  $q$  be the first and last node of  $P$  in  $T$
    - Insert sub-path  $(u,p)$  and sub-path  $(q,v)$  of  $P$  into  $T$
- ▶ **Output: Steiner tree approximation  $T$**

# MST Steiner Tree Approximation Example

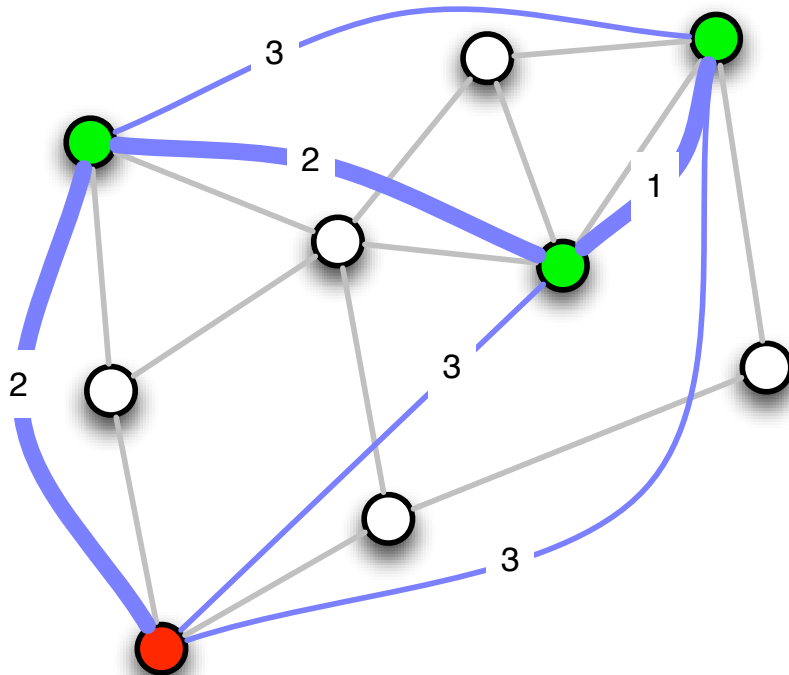


# MST Steiner Tree Approximation Example

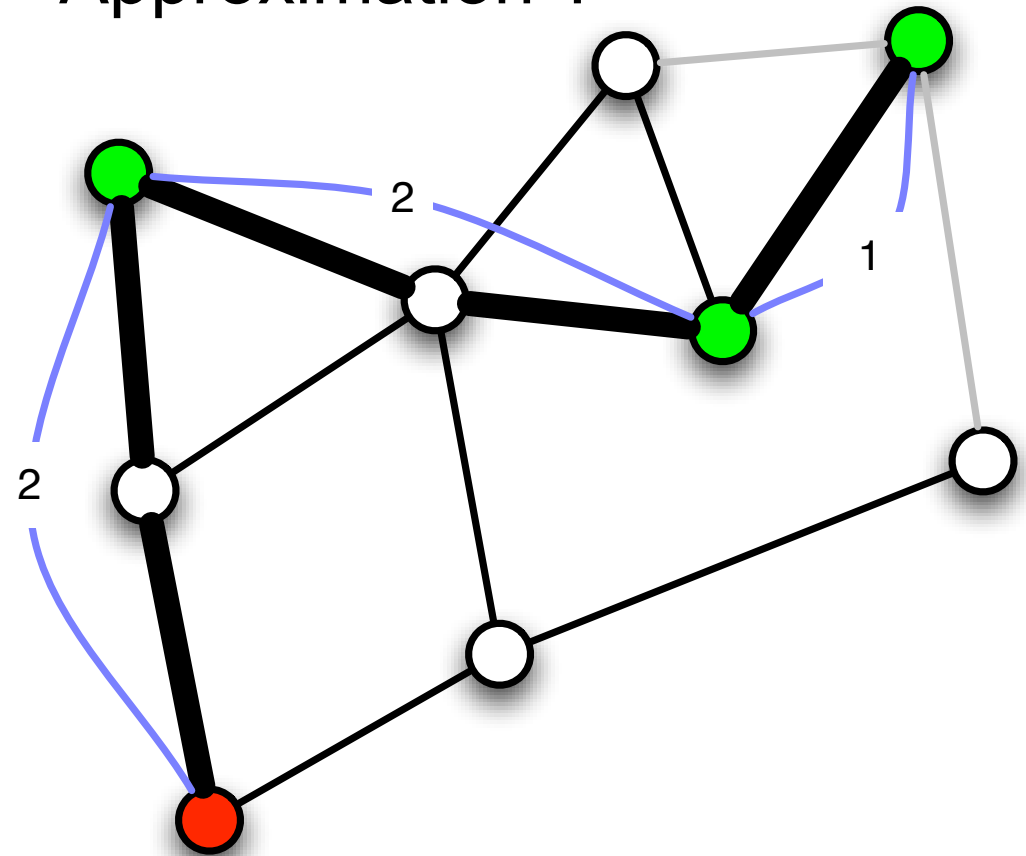


# MST Steiner Tree Approximation Example

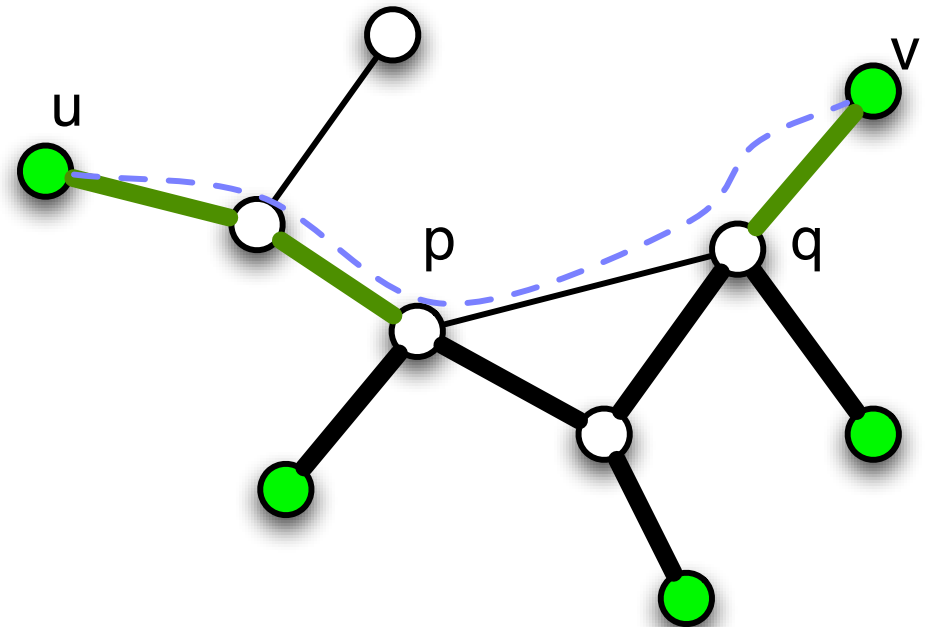
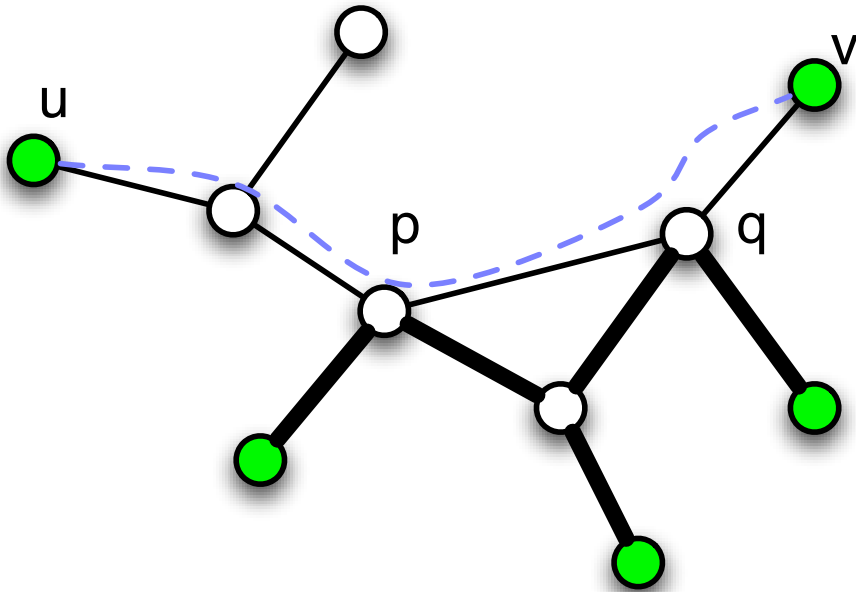
MST( $G_L$ )



Steiner Tree  
Approximation T



# Preventing Cycles



# Quality of the MST-Approximation

## ► Theorem

- The MST-Approximation constructs a tree in polynomial time. The edge sum of this tree is at most twice as large as those of the Steiner tree.

## ► Proof sketch

- A Steiner Tree without Steiner points is a factor 2 approximation of the Steiner tree
- The minimum spanning tree of  $G_L$  is the Steiner Tree without Steiner points of  $G$
- This results in approximation algorithm with factor 2

# Routing Models for Data Aggregation

## ▶ Address Centric Protocol

- each sensor sends independently towards the sink
- not suitable for (real) aggregation

## ▶ Data Centric Protocol

- Forwarding nodes can read and change messages

## ▶ Literature

- Krishnamachari, Estrin, Wicker The Impact of Data Aggregation in Wireless Sensor Networks, Proc. of the 2nd Int. Conf. on Distributed Computing Systems Workshops (ICDCSW'02)

# Energy Optimal Tree Structure

► **Given:**

- set of data sources and a sink
- communication graph  $G$

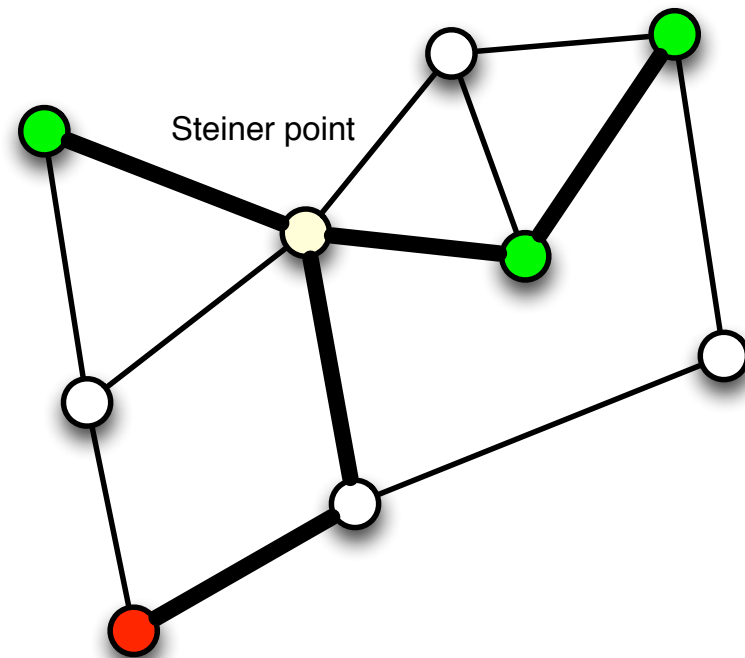
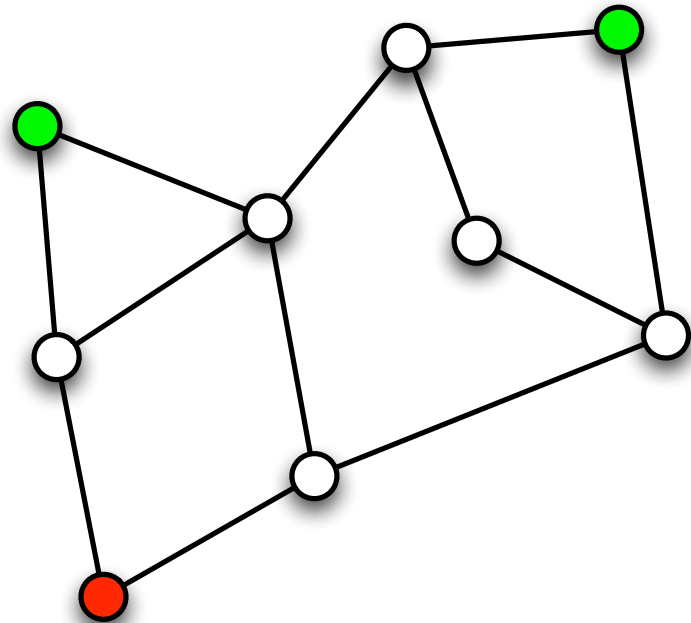
► **Compute:**

- Steiner tree  $T$ 
  - sub-graph of  $G$
  - connects all sources and sinks
  - number of edges is minimal

► **Alternative:**

- edges have an (energy) weight
- minimize the sum of edge weights

# Steiner Tree Problem



# Theoretical Bounds

► **Costs for address based Routing  $N_A$**

$$N_A = \sum_i d_i$$

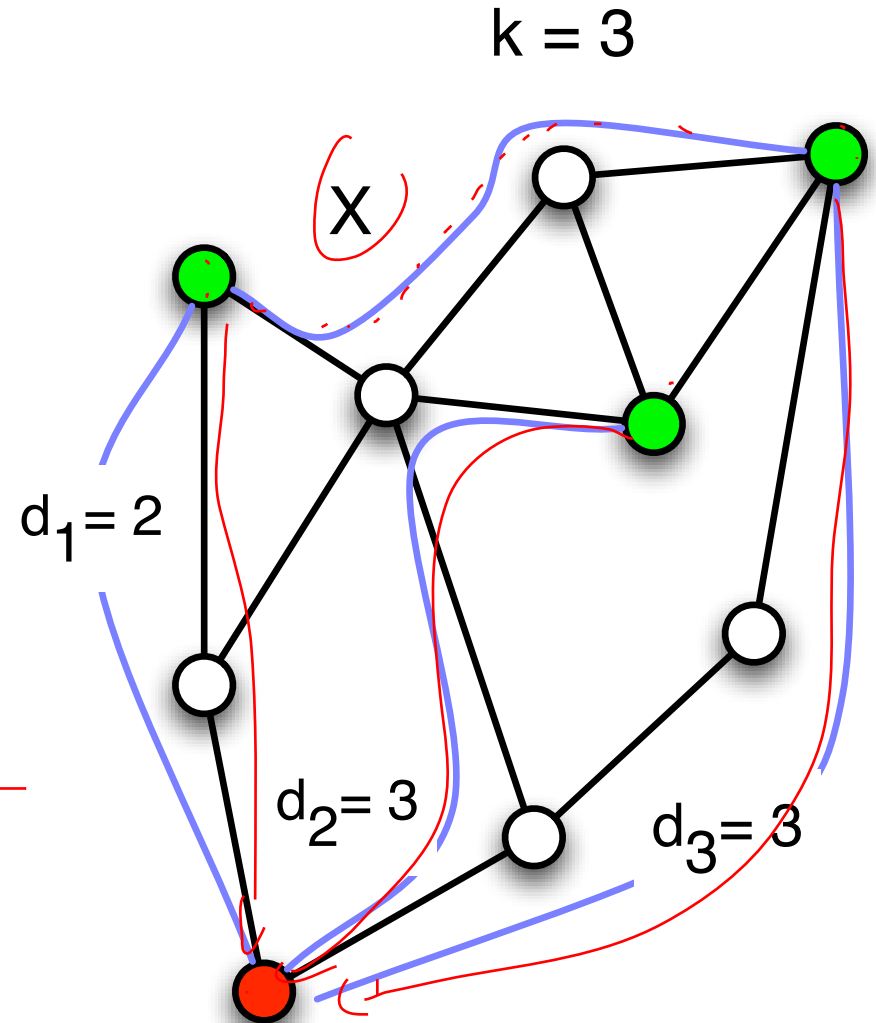
- $d_i$ : shortest distance from source  $i$  to sink  $s$

► **Cost for optimal data centric routing  $N_D$  = weight of Steiner-tree**

$$N_D \leq (k - 1)X + \min_i \{d_i\}$$

- $X$ : maximal shortest path between sources
- $k$ : number of sources

$$N_D \geq \min_i \{d_i\} + k - 1$$



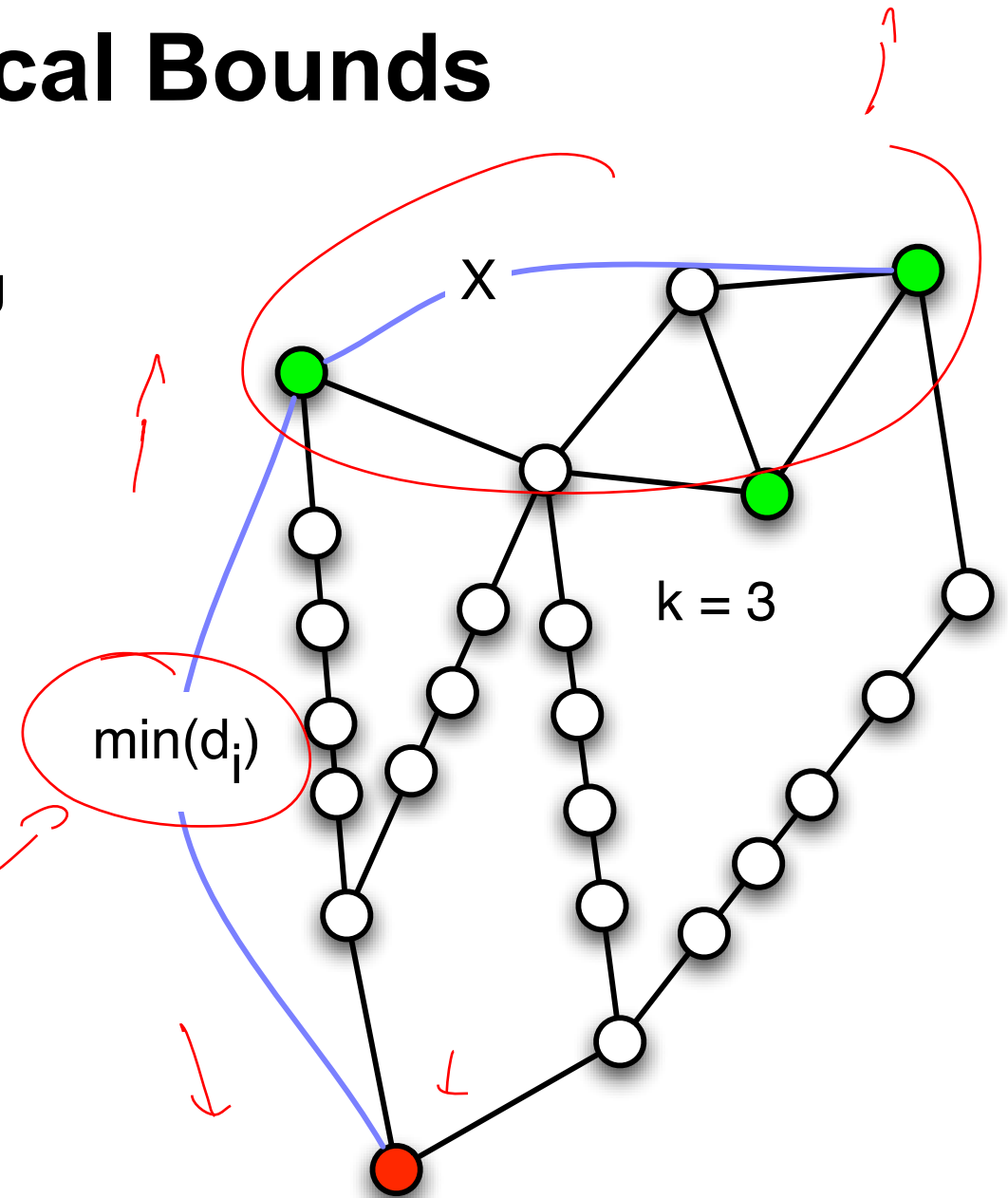
# Theoretical Bounds

- For fixed X and k and growing  $\min_i\{d_i\}$

$$\lim_{d \rightarrow \infty} \frac{N_D}{N_A} = \frac{1}{k}$$

$$N_D \sim \min(d_i) + O(1)$$

$$N_A \sim k \cdot \min(d_i)$$



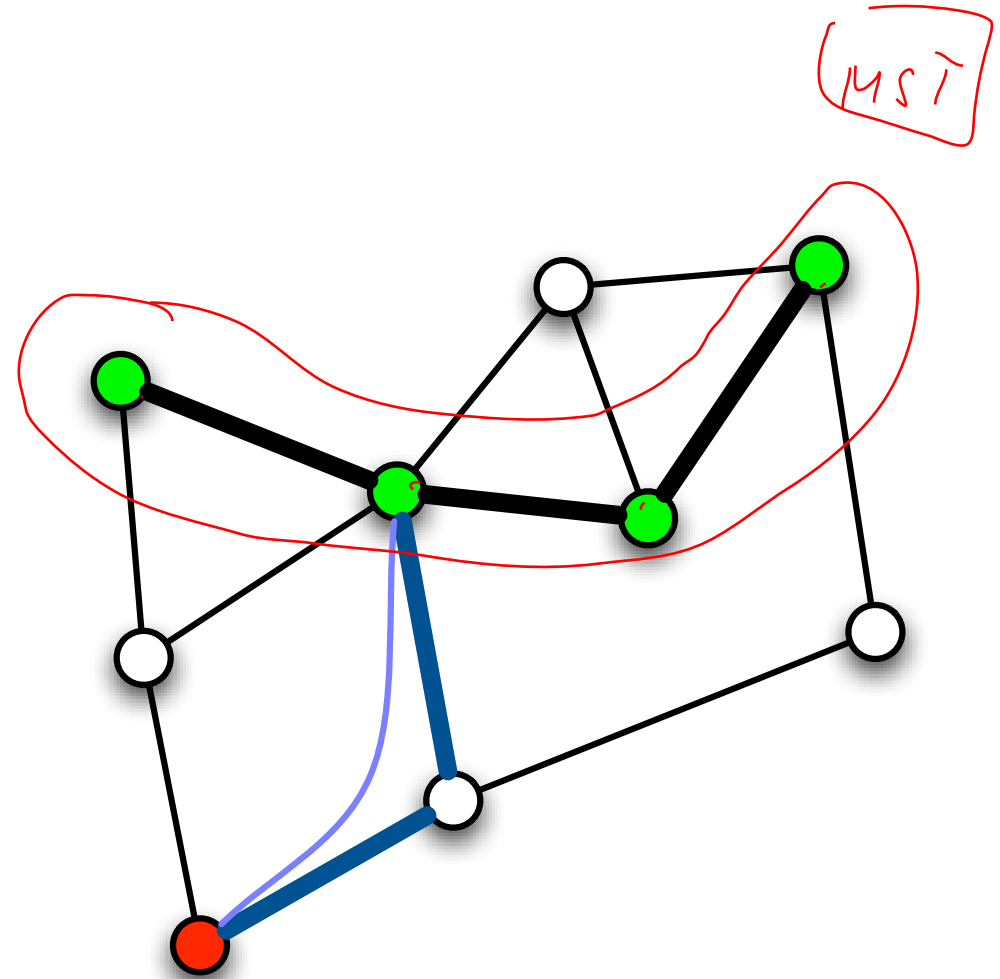
# Theoretical Bounds

## ► Theorem

- If the subgraph induced by the sources is connected, then the optimal routing can be computed in polynomial time

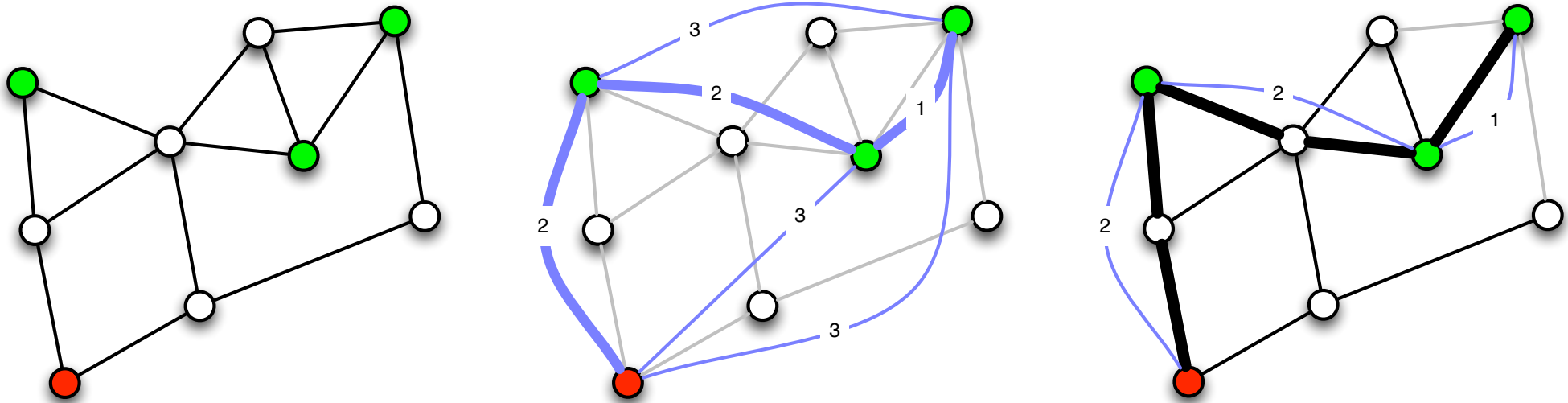
## ► Proof sketch

- Compute MST  $T$  for the sources
- Compute the shortest path from  $T$  to the sink



# Approximation Algorithm

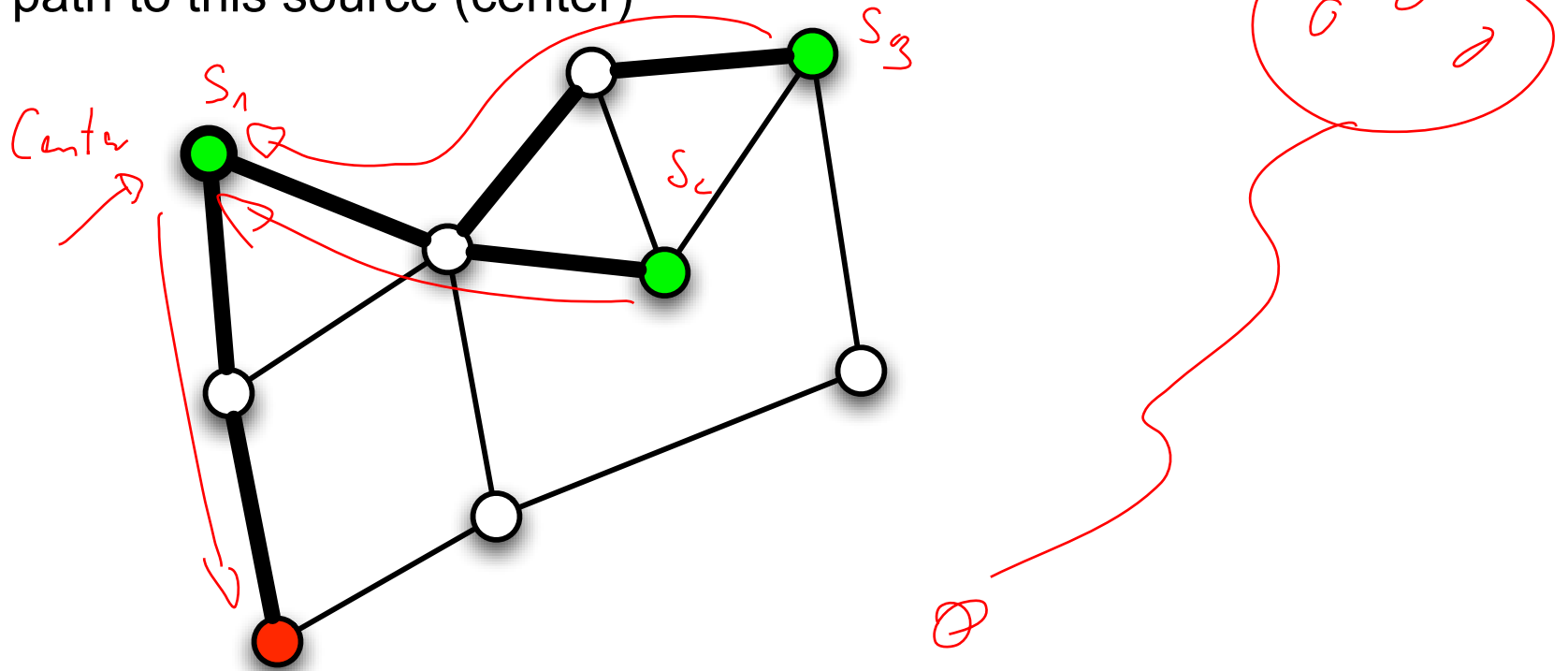
- ▶ The Steiner tree approximation algorithm (of the last lecture) cannot be implemented efficiently in a WSN



# Suboptimal Aggregation

## ► **Center at Nearest Source (CNS)**

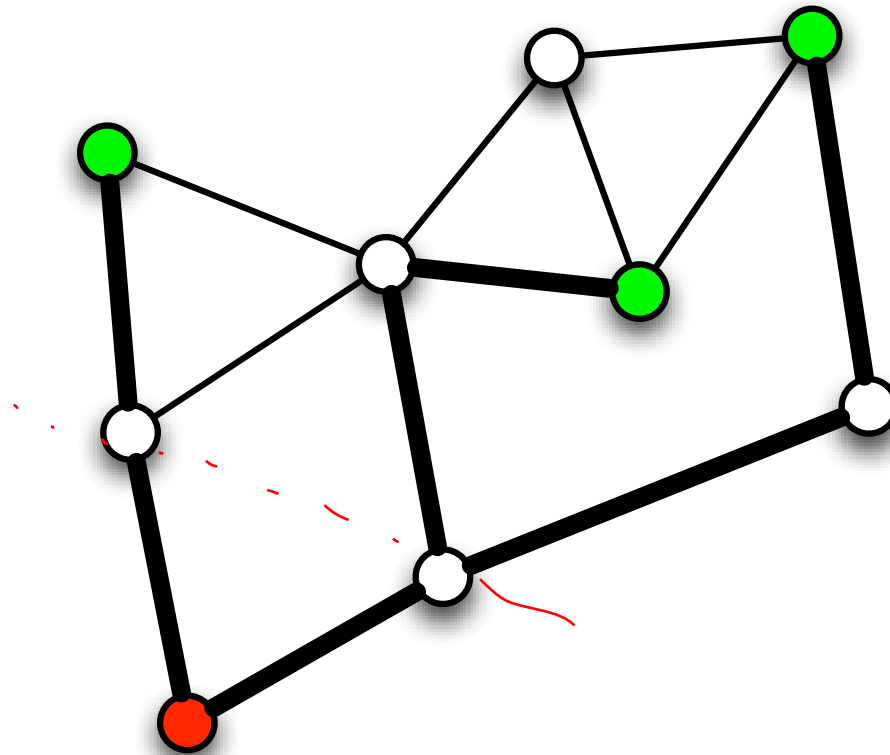
- Data source closest to the sink collects all information
- All other sources send the information on the shortest path to this source (center)



# Suboptimal Aggregation

## ► Shortest Paths Trees (SPT)

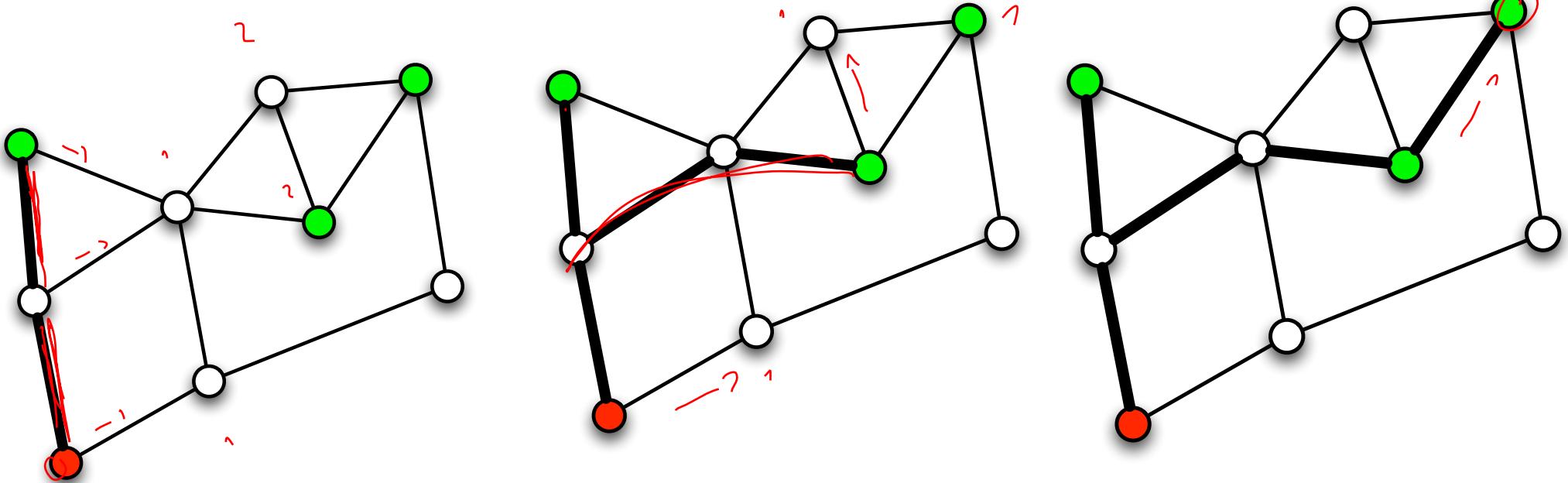
- Set of all shortest paths from the sources to the sink



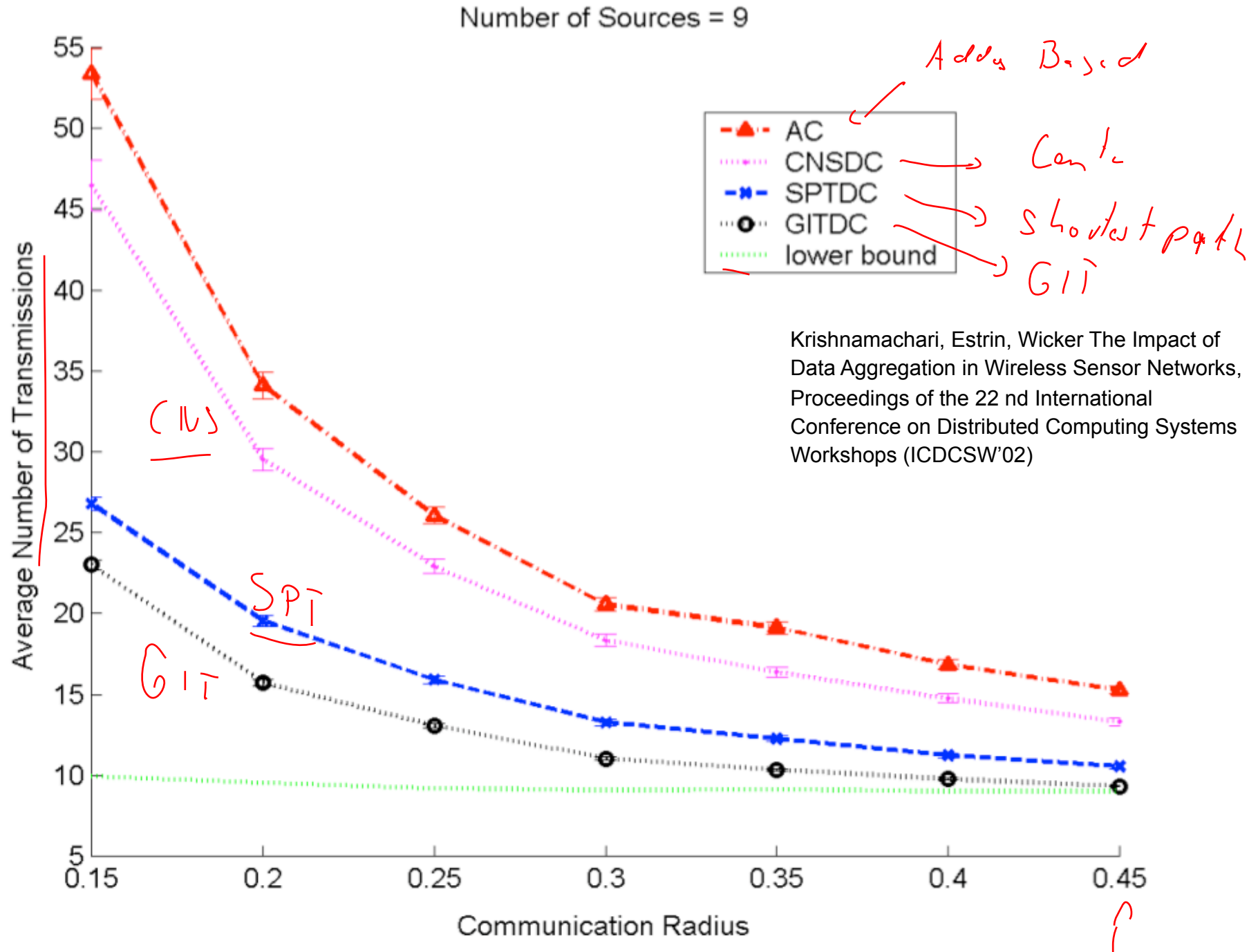
# Suboptimal Aggregation

## 1) Greedy Incremental Tree (GIT)

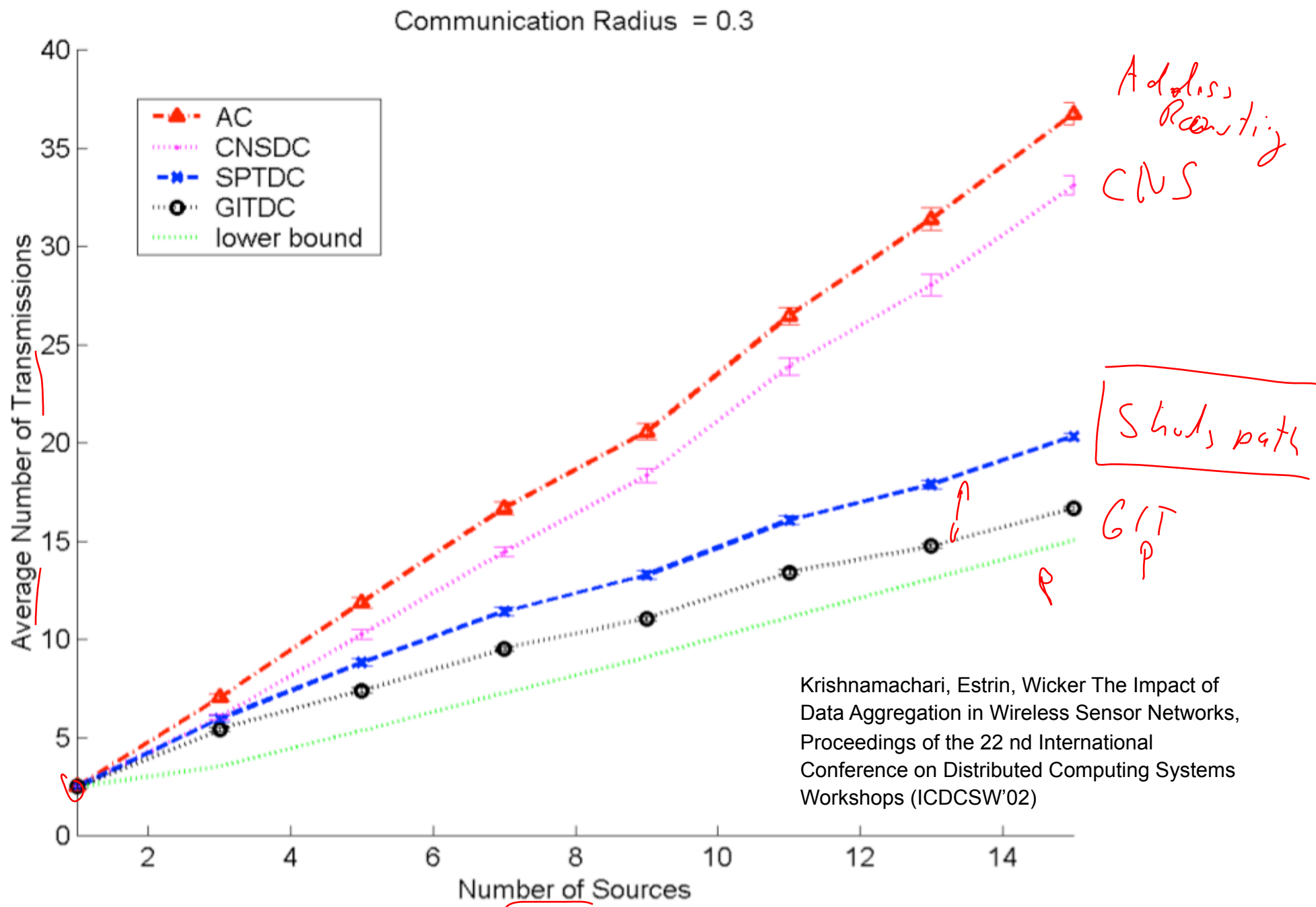
- Select the shortest path between the data source, closest to the sink, and the sink
- Select successively the closest node to the tree and the shortest path to any of the tree nodes



# Energy Saving by Data Aggregation



# Energy Saving by Data Aggregation



# Probabilistic Counting for Data Aggregation

- ▶ **Hard problems for Data Aggregation**
  - Counting of different elements in a multiset
  - Computation of Median
- ▶ **Exact computation needs complete knowledge**
  - therefore we compute approximations
- ▶ **Main Technique**
  - probabilistic counting
    - „Counting by Coin Tossings“, Philippe Flajolet, ASIAN 2004
  - probabilistic sampling
    - „A note on efficient aggregate queries in sensor networks“, Boaz Patt-Shamir, Theoretical Computer Science 370 (2007) 254–264

# Morris Counting Method

## ► Literature

- Morris, R. Counting large numbers of events in small registers. Communications of the ACM 21, 10 (1977), 840–842.

## ► Task

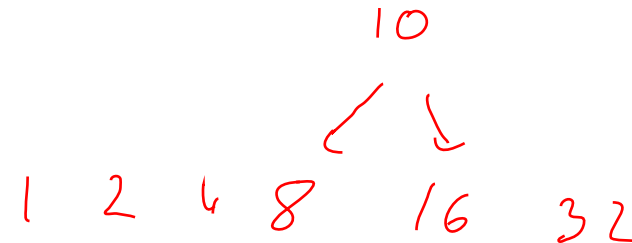
- Count to  $n$  using only  $\log \log n$  bits

## ► Motivation

- In the early days very little memory was available

## ► Algorithm

- $k := 1$
- For each counting event
  - flip  $k$  fair coins  $C_i$  in  $\{0,1\}$
  - If  $C_1 = C_2 = C_3 \dots = C_k = 0$  (i.e. with probability  $2^{-k}$ )
    - \* then  $k := k+1$
- Return  $2^{k-1}$



# Analysis of Morris Counting Algorithm

- The probability to increase a counter from  $k$  to  $k+1$ :

-  $2^{-k}$

- Probability to increase a counter after  $i$  steps:

-  $(1-2^{-k})^{i-1} 2^{-k}$

- Expected Time to increase a counter from  $2^k$  to  $2^{k+1}$

- let  $q = 1-2^{-k}$

$$\sum_{i=1}^{\infty} i(1-2^{-k})^{i-1} 2^{-k}$$

$1-q = 2^{-k}$

$$\begin{aligned} &= 2^{-k} \sum_{j=1}^{\infty} j q^{j-1} \\ &= 2^{-k} \sum_{j=1}^{\infty} \sum_{i=j}^{\infty} q^{i-1} \\ &= 2^{-k} \sum_{j=1}^{\infty} q^{j-1} \sum_{i=0}^{\infty} q^i \\ &= 2^{-k} \sum_{j=0}^{\infty} q^j \left[ \frac{1}{1-q} \right] \\ &= 2^{-k} \left( \frac{1}{1-q} \right)^2 = 2^{-k} \cdot 2^k \cdot 2^k \\ &= 2^k \end{aligned}$$

# Does Morris help us?

## ► Morris Counter algorithm

- in the expectation counts correctly up to a factor of two
- with only  $\log \log n + O(1)$  bits memory

## ► However:

- Counting is an easy problem for WSN
  - using  $\log n$  bits
- Does it help us to count sets?

# Counting Elements in Multisets

## ► A Multiset

- an unordered collection of elements, which might occur more than once
- e.g. {mouse, mouse, cat, dog, cat}

## ► Task

- Compute the number of different elements in a multiset with little memory (message size)

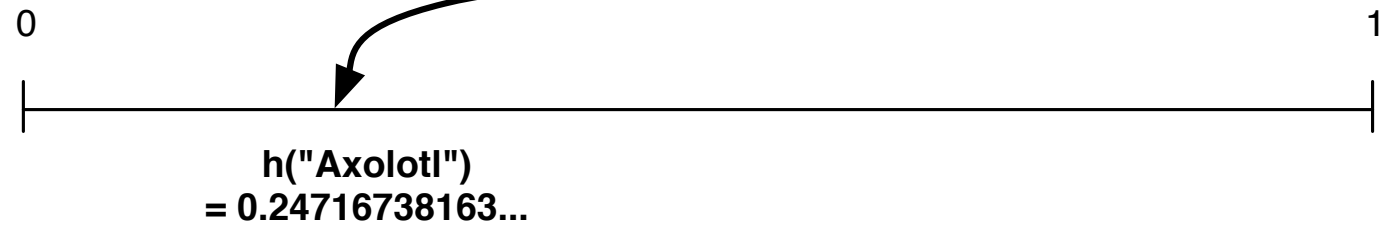
## ► Solution

- Consider a perfect hash function
- Count elements in intervals

# Flajolet's Counting Algorithm

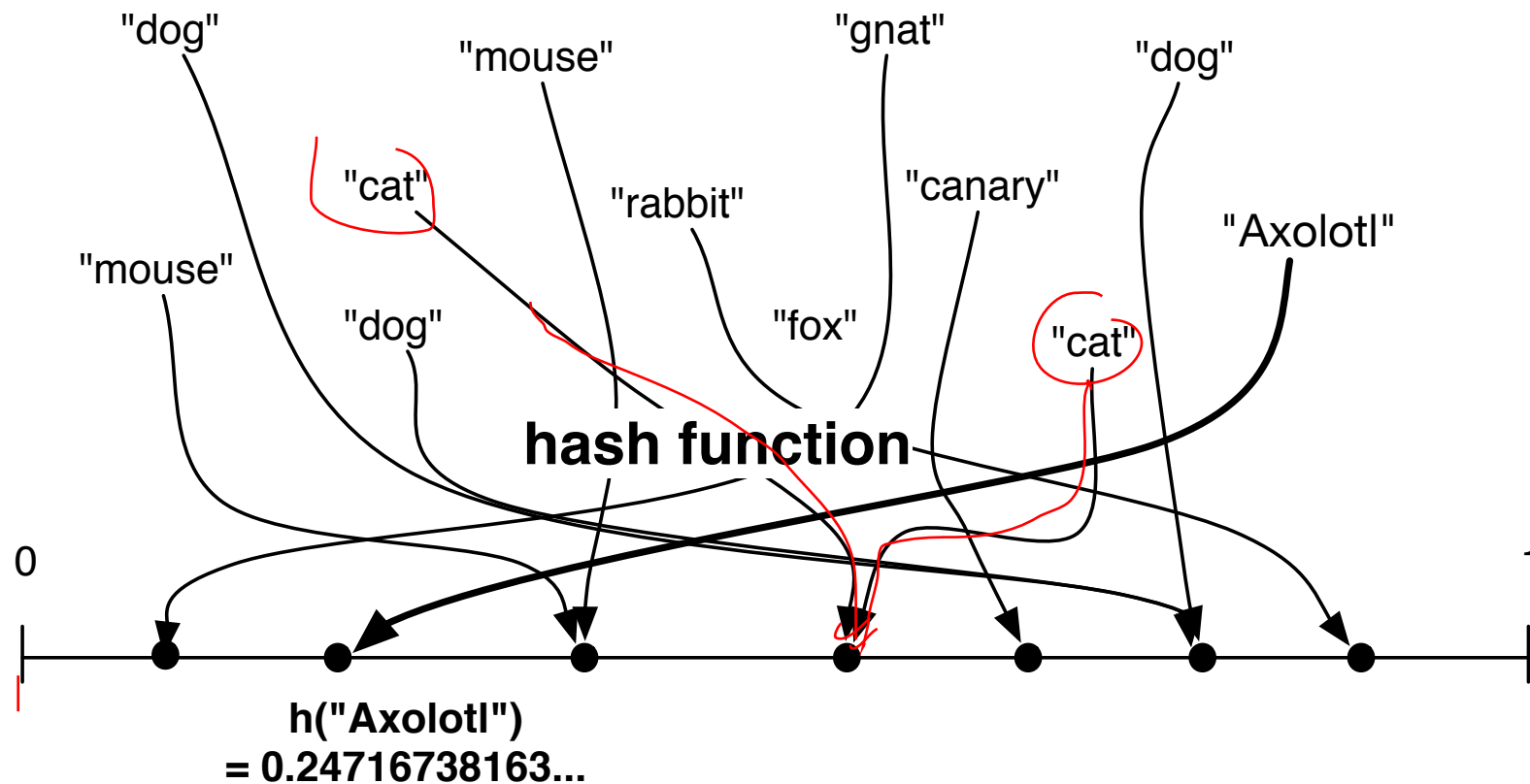
"fly" "mouse" "fly" "rabbit" "mouse"  
"dog" "mouse" "gnat" "dog"  
"cat" "rabbit" "canary" "Axolotl"  
"mouse" "dog" "fox" "cat"

# How to Compute a Hash Function



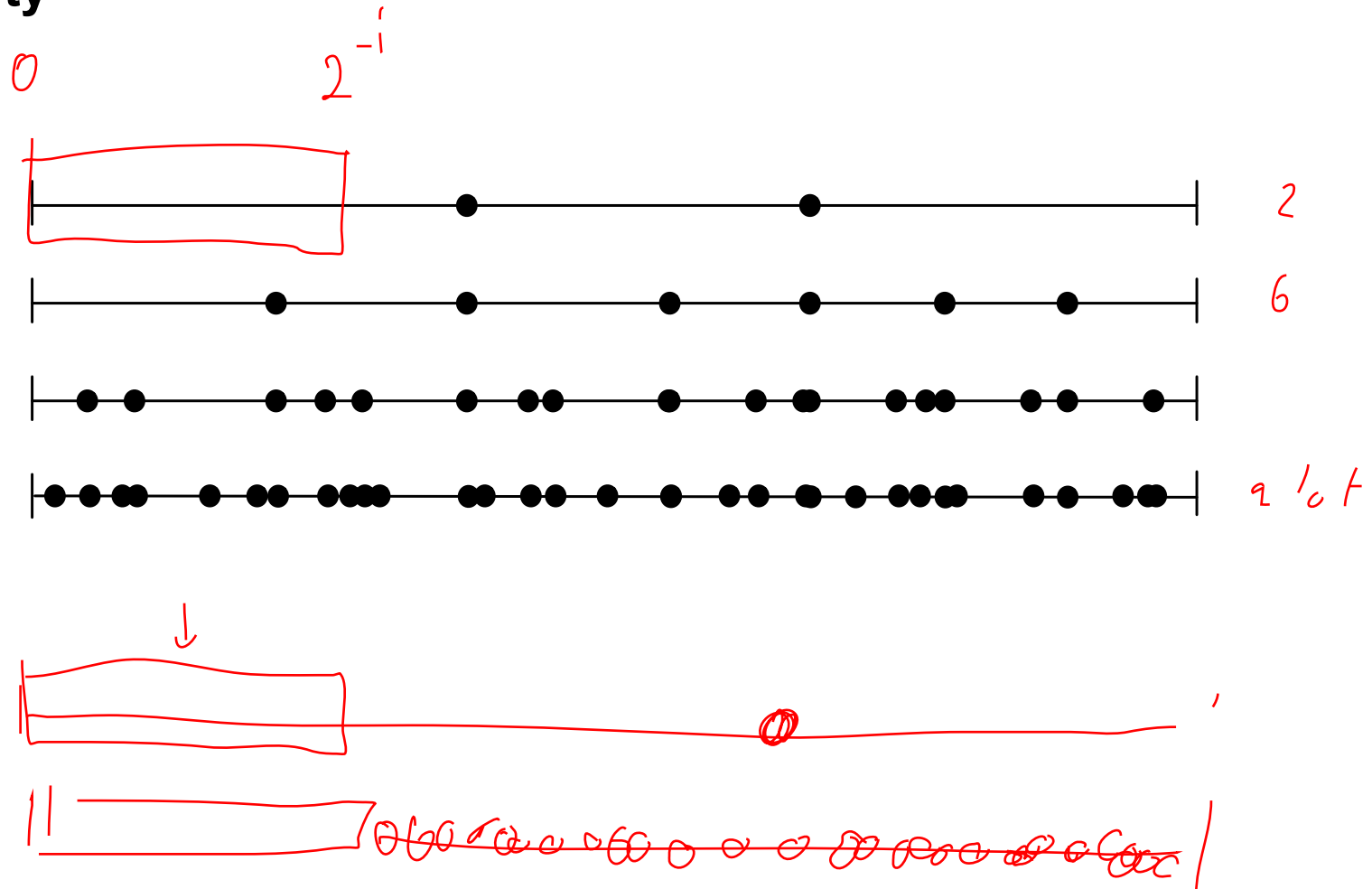
Text	A x o l o t l
ASCII	65, 120, 111, 108, 111, 116, 108
Binary Digits	1000001 1111000 1101111 ...
some cryptographic function, e.g. AES("1000001111...")	0011111101000110010111...
hash value (binary)	0.0011111101000110010111...
hash value (decimal)	0.24716738163 ...

# Flajolet's Counting Algorithm



# Approximate Counting

- ▶ The density of good hash functions is a global property



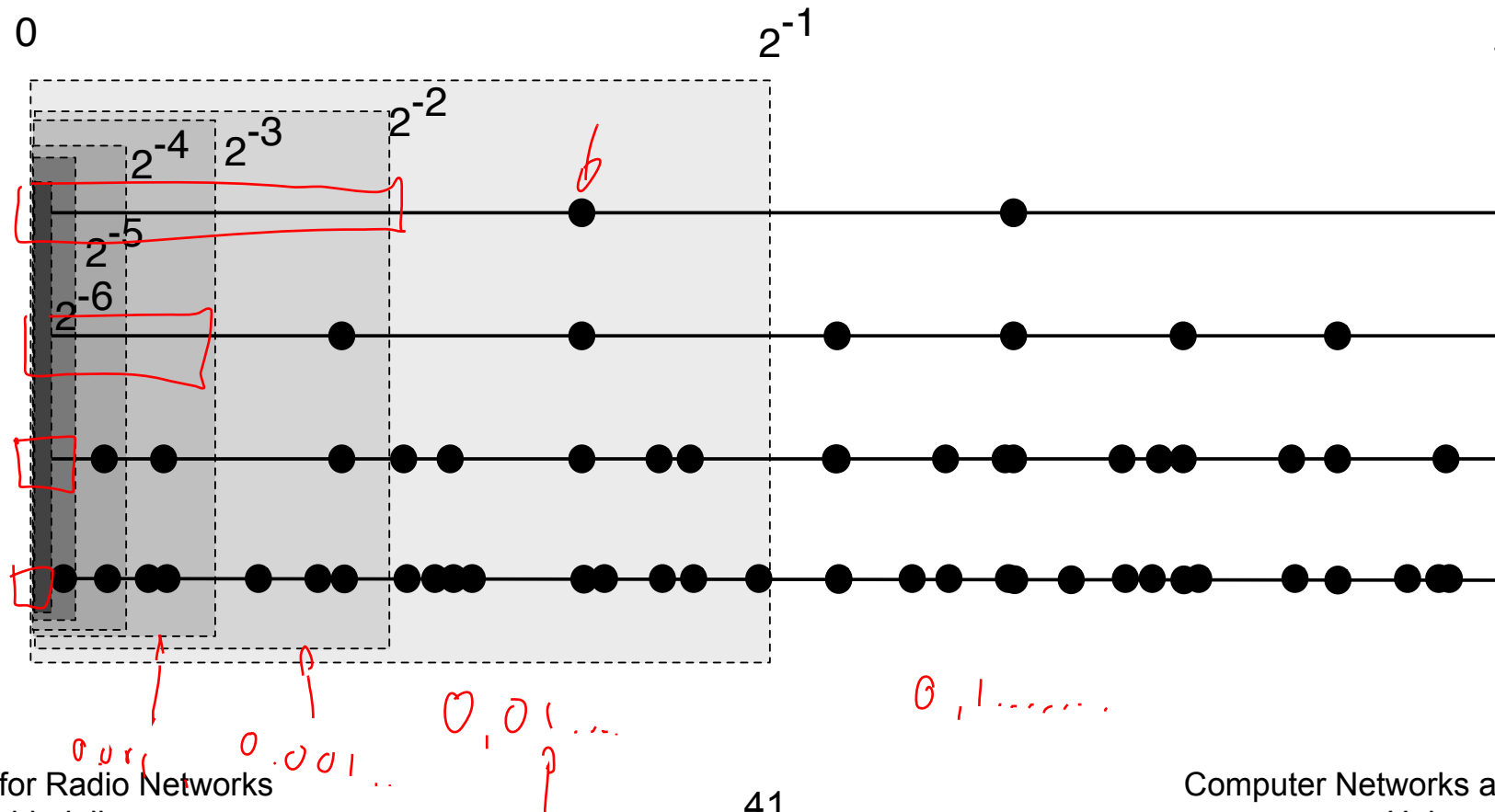
# Choose Smallest Non-empty Interval $[0, 2^k]$

## ► Analogous to Morris counting algorithm

- Truncate hash value to  $L > \log n$  digits
- Store only the term  $k$  from

$$2^{-4} \cdot n = 1$$

$$n = 2^4$$



# Counting Algorithm

- Flajolet, P., and Martin, G. N. Probabilistic counting algorithms for database applications. Journal of Computer and System Sciences 31, 2 (Oct. 1985), 182–209.

**function** hash( $x$ : records): scalar range  $[0 \cdots 2^L - 1]$ ,

$$y = \sum_{k \geq 0} \text{bit}(y, k) 2^k.$$

$$\rho(y) = \begin{cases} \min_{k \geq 0} \text{bit}(y, k) \neq 0 & \text{if } y > 0 \\ L & \text{if } y = 0. \end{cases}$$

**for**  $i := 0$  **to**  $L - 1$  **do**  $\text{BITMAP}[i] := 0$ ;

**for all**  $x$  **in**  $M$  **do**

**begin**

$\text{index} := \rho(\text{hash}(x))$ ;

**if**  $\text{BITMAP}[\text{index}] = 0$  **then**  $\text{BITMAP}[\text{index}] := 1$ ;

**end**;

$$L > \lg n$$

$$L$$

$$01101101101$$

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 1 & & & & \\ 0 & 1 & 2 & 3 & 4 & & & & \\ \downarrow & \downarrow & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \end{array}$$

# Flajolet's Probabilistic Counting

## ► Probabilistic Counting

- using m such counts the precision can be improved

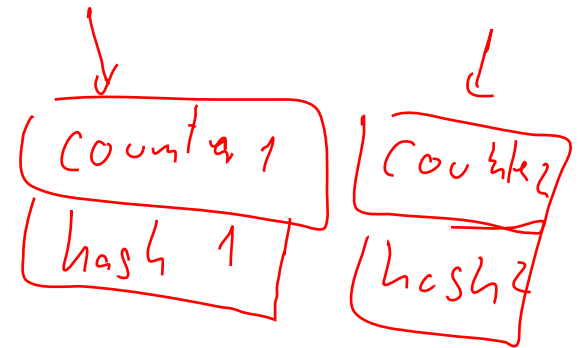
## ► Idea:

- Compute m counters
- Average over all counters

## ► Performance

- Memory:  $O(m \log \log n)$  for each message
- Precision:

$$O\left(\frac{1}{\sqrt{m}}\right)$$



# Efficient Data Aggregation of the Median

## ► Median

- Given  $n$  sensor values  $X = \{x_1, x_2, \dots, x_n\}$
- Compute  $\text{median}(X) := y_{\lfloor n/2 \rfloor}$ 
  - $\{x_1, \dots, x_n\} = \{y_1, \dots, y_n\}$
  - $y_1 \leq \dots \leq y_n$

## ► Solution

- Compute random element
- Perform binary search for median
  - by request random element in smaller intervals
  - and count elements smaller than intermediate result

# Data Aggregation of a Random Element

## ► Given

- n inputs  $x_1, \dots, x_n$
- in a sensor field

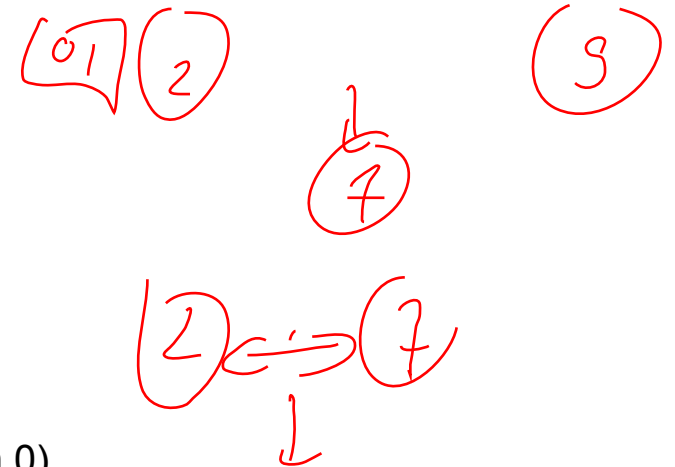
## ► Compute

- $x_i$ , such that i is uniformly chosen from  $\{1, \dots, n\}$

## ► Aggregation Algorithm (on a tree)

- Receive element count  $n_j$  from branch j
- Receive random element  $x_j$  from branch j
- $n_0$ : number of own elements
- $x_0$ : uniformly chosen element (own elements are branch 0)
- Choose element x from branch j with probability  $\frac{n_j}{\sum_{i=1}^k n_i}$
- Compute  $n = \sum_{i=1}^k n_i$
- Send random element x and counter n to the next higher tree node

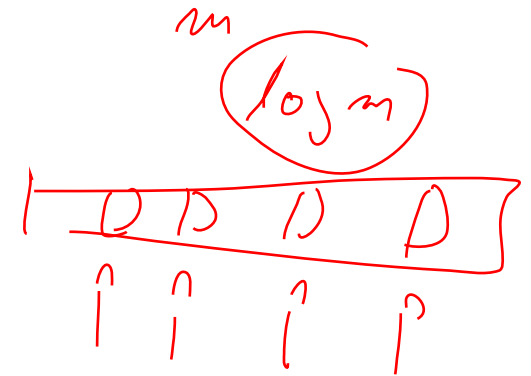
0 1 2 3 ... 9



# Computing the Median via Binary Search

## ► Median Algorithm

- $n > 3$  is even
- $x_{\text{start}} = \min\{X\}$ ,  $x_{\text{end}} = \max\{X\}$ ,  $x_{\text{random}} := \text{random element}$
- $m = \text{count of elements} \leq x_{\text{random}}$
- While  $m \neq n/2$  do
  - If  $m < n/2$ 
    - \*  $x_{\text{start}} := x_{\text{random}}$
    - \*  $x_{\text{random}} := \text{random element form } (x_{\text{start}}, x_{\text{end}})$
  - else
    - \*  $x_{\text{end}} := x_{\text{random}}$
    - \*  $x_{\text{random}} := \text{random element form } (x_{\text{start}}, x_{\text{end}})$
- od
- Return  $x_{\text{random}}$



# Expected Run-Time

- ▶ **Expected number of iterations of the Median algorithm is  $O(\log n)$** 
  - if  $n$  is the number of elements
- ▶ **Expected overall time:  $O(D \log n)$** 
  - if  $D$  is the network diameter
- ▶ **The expected number of messages is  $O(n \log n)$ ,**
  - if  $n$  is the number nodes as well
- ▶ **The size of the messages is bounded by  $O(s)$** 
  - if the size of the elements is  $s$

# Analysis of Expected Running Time

- ▶  $T(n)$ : Expected running time to find median in interval of size  $n$ 
  - Random element is chosen with probability  $1/n$

$$\begin{aligned} T(n) &= \frac{1}{n} \left( \sum_{k=1}^n T(k-1) + T(n-k) \right) + O(D) \\ &= \frac{1}{n} \sum_{k=1}^n T(k-1) + O(D) \\ &= O(D \log n) \end{aligned}$$



ALBERT-LUDWIGS-  
UNIVERSITÄT FREIBURG

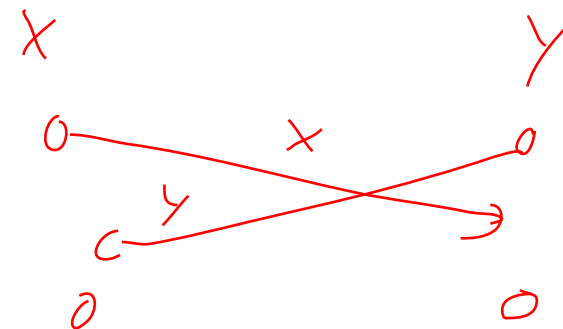
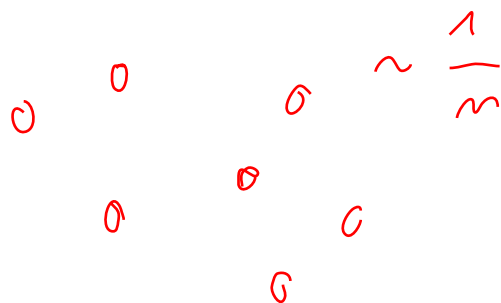
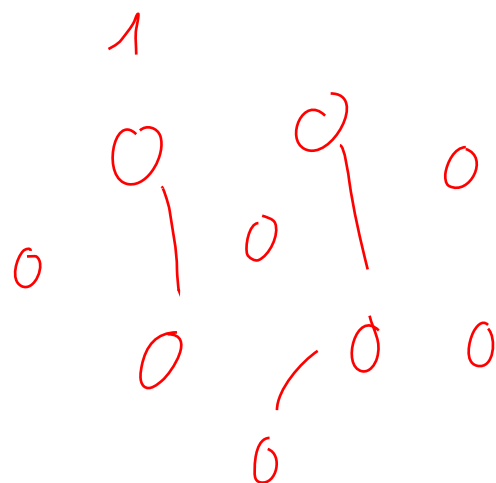
# Algorithms for Radio Networks

## Data Aggregation

University of Freiburg  
Technical Faculty  
Computer Networks and Telematics  
Christian Schindelhauer



# Population Protocol

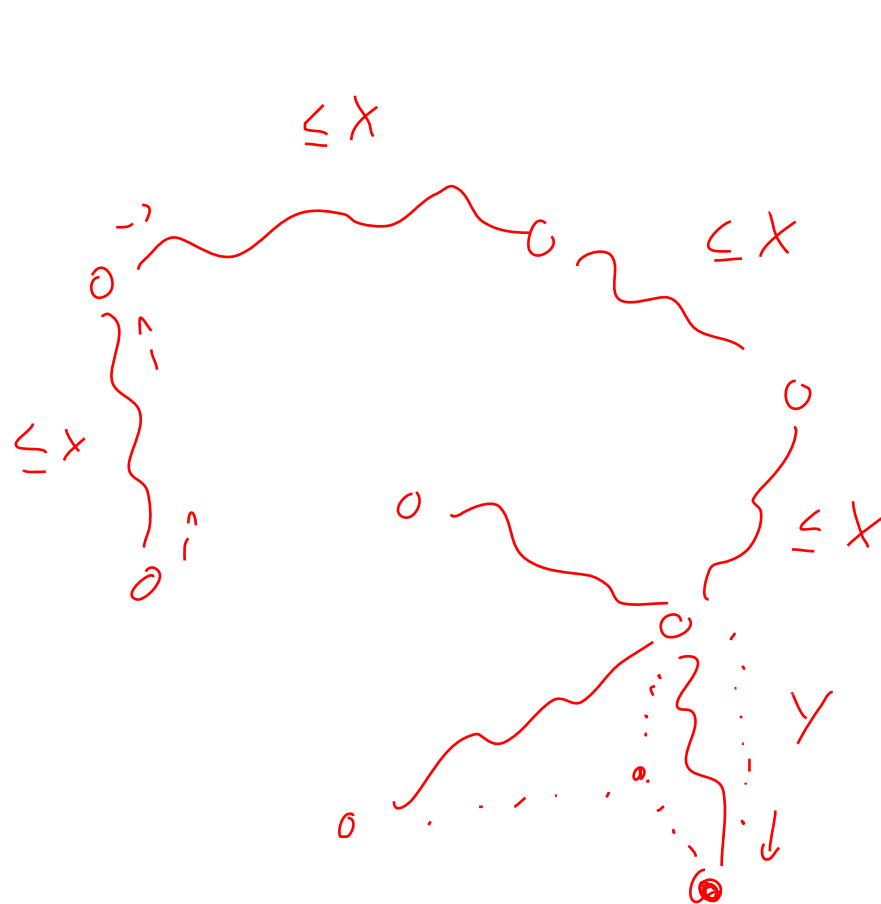


$$\frac{x+y}{2}$$

$$\frac{x+y}{2}$$

A	B	C
0	0	0
1	0	0
0.5	0.5	-
	0.25	0.25

$$0.375 \quad 0.25 \quad 0.375$$



terminal  
 $k$ : # nodes

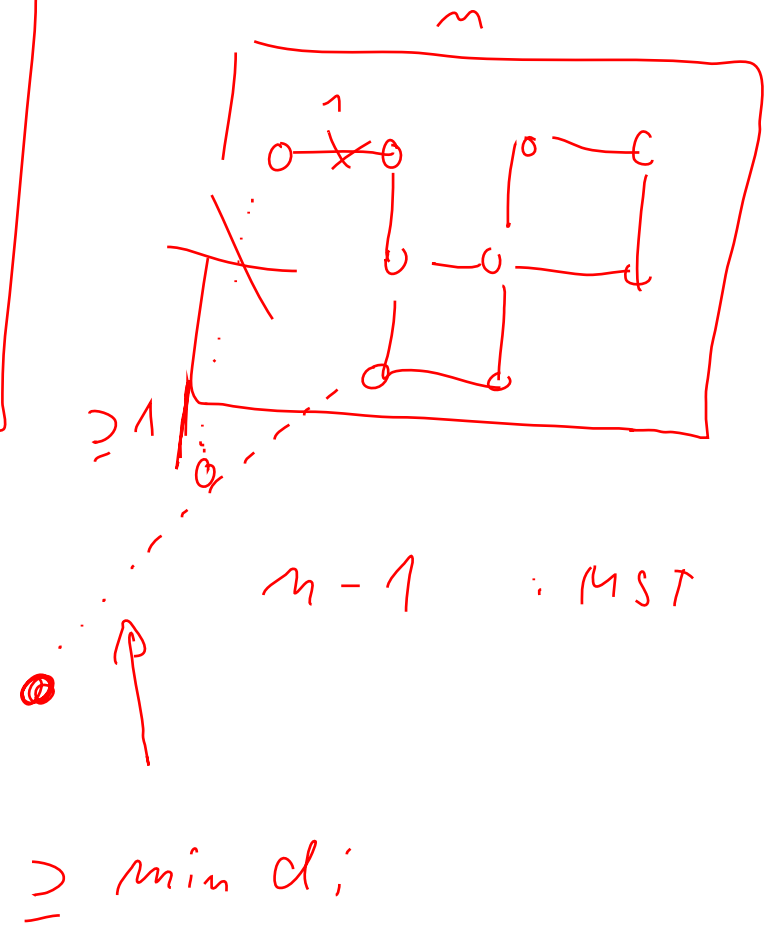
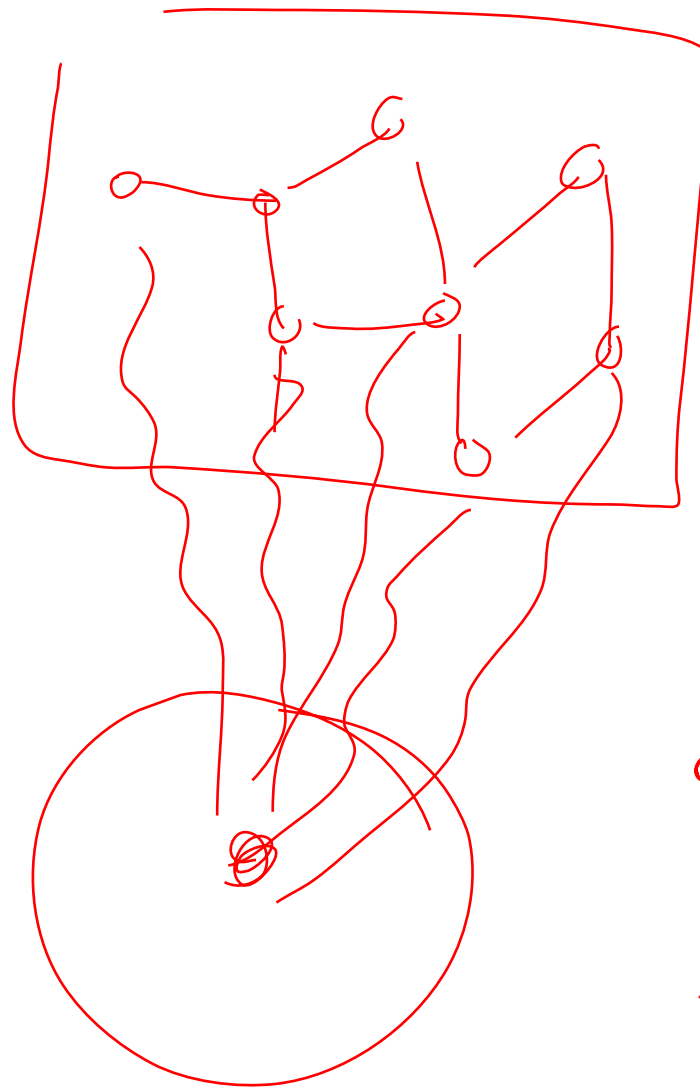
$$(k-1) \cdot X$$

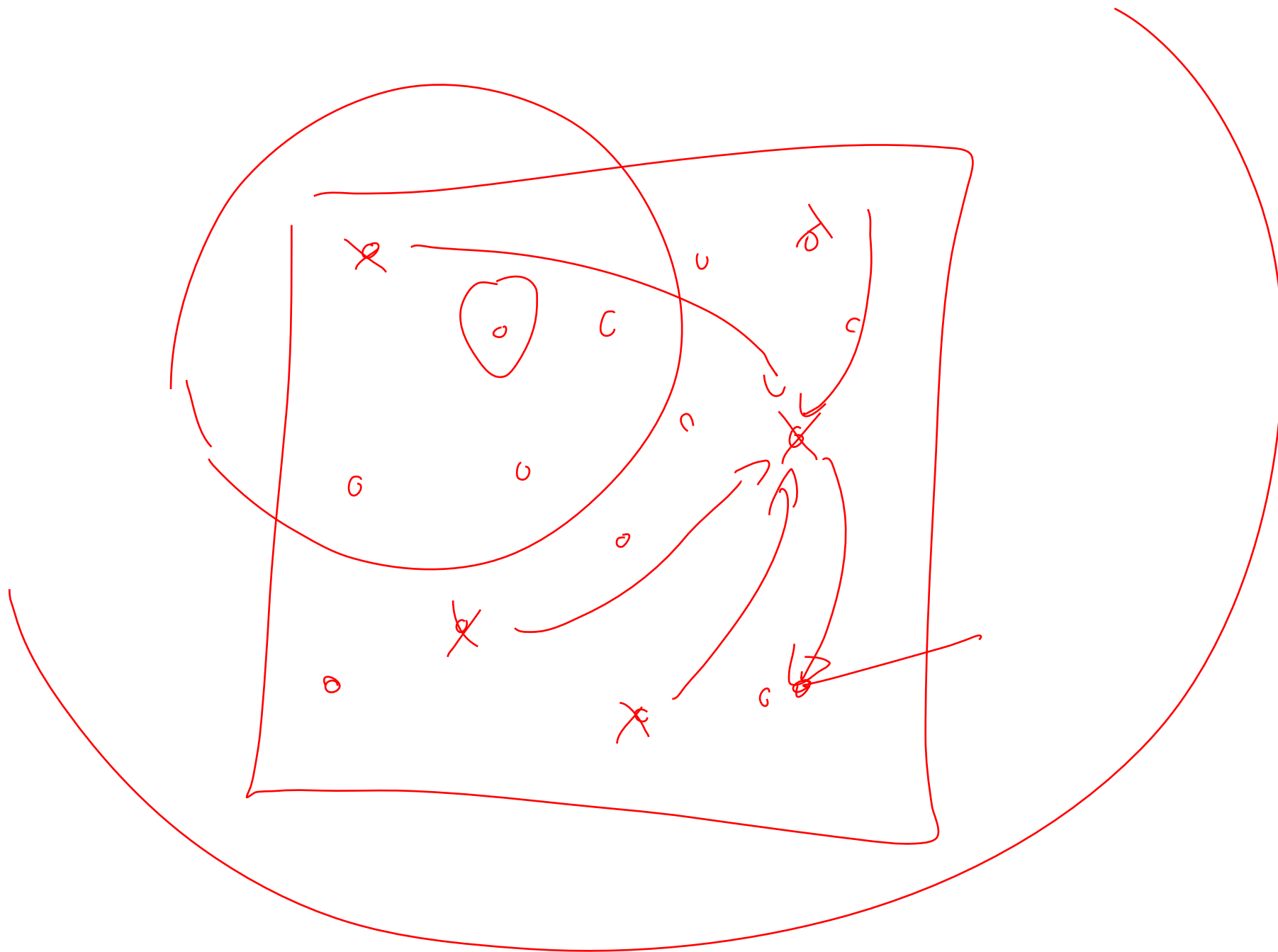
$\vee$

$N_0$

$\vee$

$\gamma$







Breudcker!  
Kubelz

~~9~~ → 10

Unary encoding  
Storage size  
 $n$

$n \rightarrow$

$n$

$\rightarrow$

$\log_{10} n$

decimal

?

$n$

$\rightarrow$

$\log_2 n$

binary

$n \rightarrow \log \log n$

Morris

$$\frac{\log_{10} n}{\log_2 n} = \frac{\ln 2}{\ln 10}$$

$$1 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 3$$
$$\frac{1}{2} \quad \frac{1}{2} \quad \frac{3}{4} \quad \frac{1}{4}$$

$$E[X] = \sum_{x=1}^{\infty} P[X=x] \cdot x = \sum_{x=1}^{\infty} x \cdot \left(\frac{1}{2}\right)^x = 2$$

$f: 1 \rightarrow 2$

$$P[X=1] = \frac{1}{2}$$

$$P(X=2) = \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2}$$

$$P(X=3) = \frac{1}{8}$$

$$\sum_{i=1}^{\infty} i \cdot 2^{-i} = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 4 \cdot \frac{1}{16} + \dots$$

$$= \left( \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots \right) + \left( \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots \right) + \frac{1}{8} + \frac{1}{16} + \dots$$

$$= 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots$$

$$= \boxed{2}$$

$0.1_2$   
 $0.01_2$   
 $0.001_2$

$$0.111111_2 = 1$$

$$k = 10$$

$$\sum_{i=0}^{\infty} q^i = \frac{1}{1-q}$$

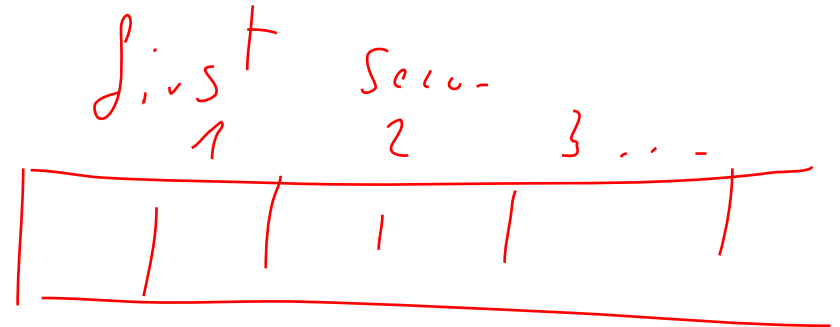
$$\frac{1}{1024}$$

$$\frac{1023}{1024}$$

$$q < 1$$

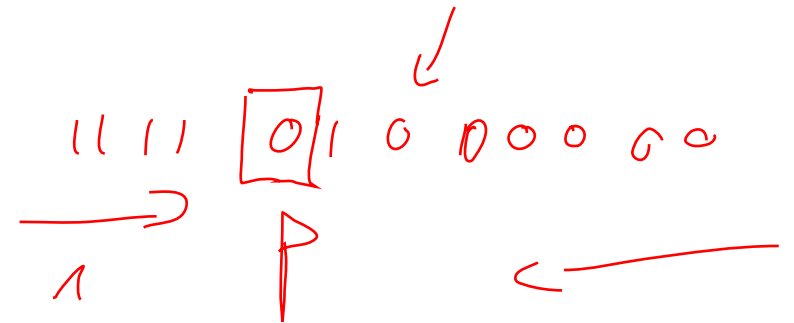
$$\begin{aligned} (1-q) \left( \sum_{i=0}^{\infty} q^i \right) &= (1 + q + q^2 + q^3 + \dots) (1-q) \\ &= 1 + q + q^2 + q^3 + \dots \\ &\quad - q - q^2 - q^3 - q^4 - \dots \\ \hline &= 1 \end{aligned}$$

$$\begin{array}{r|l}
 \rightarrow & 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \\
 & \cup \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \\
 \rightarrow & 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \\
 & 0 \ 0 \ 1 \ 0 \ 0 \\
 \rightarrow & 1 \ 0 \ 0 \ 0
 \end{array}$$



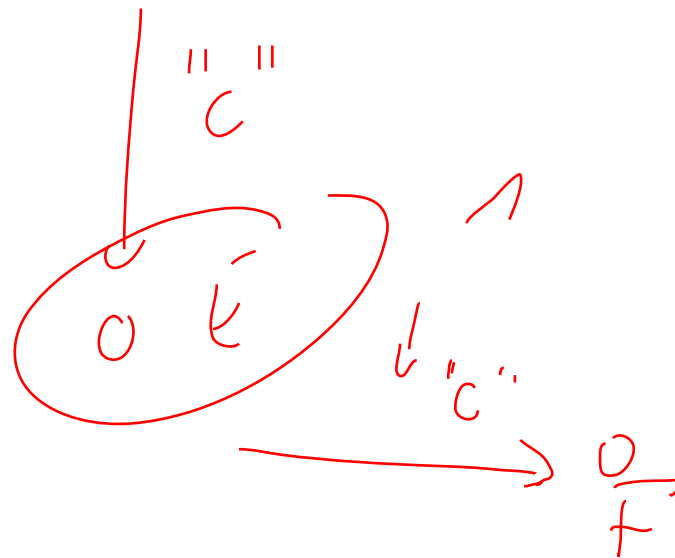
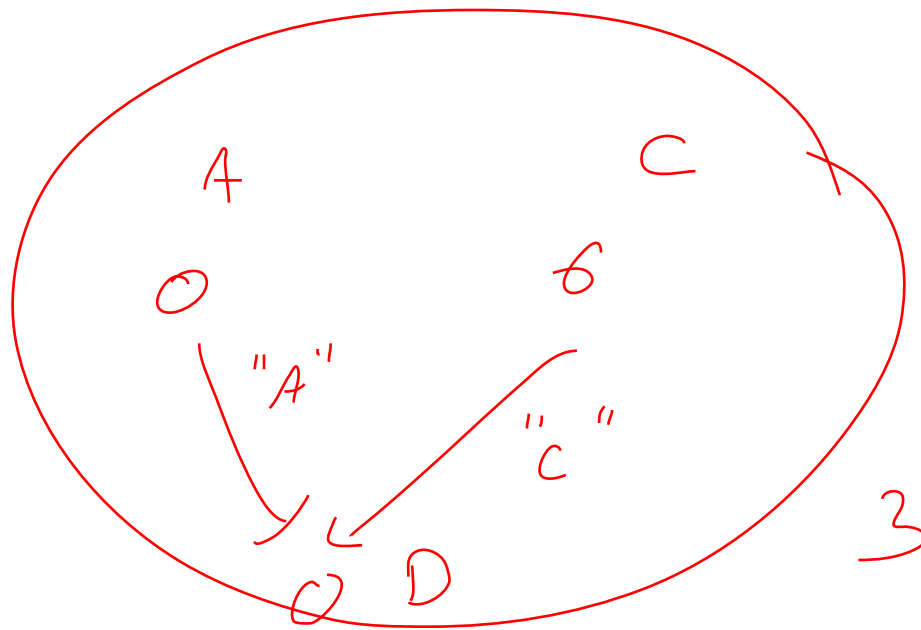
50% 1 at the first pos

$\frac{1}{4} = 25\%$  1 at the sec -



$$X_1, X_2, \dots, X_n$$

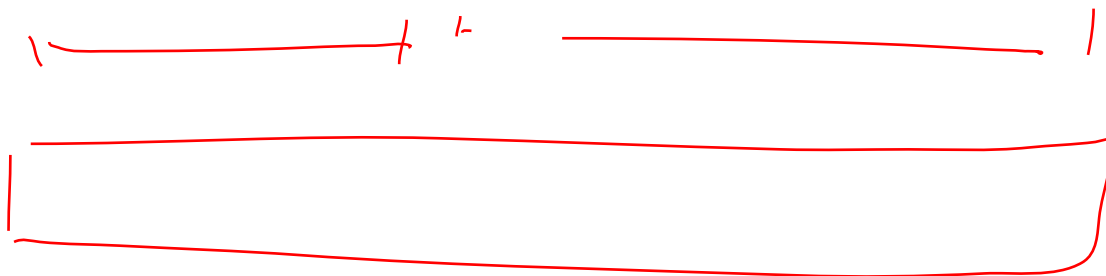
$$\frac{\sum X_i}{n} \rightarrow E[X]$$



$$\begin{aligned} "F" &= \frac{1}{5} \\ "C" &= \frac{3}{5} \end{aligned}$$

$$"E" = \frac{1}{4}$$

$$"C" = \frac{3}{4} \quad 11$$



$$\frac{1}{n}$$